UNIVERSIDAD CARLOS III DE MADRID

TRABAJO FIN DE GRADO

GENERALIZACIÓN DEL CÁLCULO DE AGARRES EN EL SIMULADOR SYNGRASP

GRADO EN TECNOLOGÍAS INDUSTRIALES

Autor: Iván Gutiérrez Martín.

Tutor: David Álvarez Sánchez.

LEGANÉS, MADRID

SEPTIEMBRE 2014

A todas las personas que

Me han ayudado a poder estar aquí

Resumen

El objetivo de este trabajo de fin de grado es la comprensión, utilización y ampliación de una *toolbox* de investigación, SynGrasp (que se encuentra en estado de desarrollo) diseñada para el software matemático MATLAB.

Esta *toolbox* es utilizada para calcular y analizar el agarre entre distintas manos (robóticas o humanoides) y objetos.

Para cumplir el objetivo, el trabajo se ha centrado en tres puntos clave:

- La incorporación a esta *toolbox* de un modelo de mano que no está recogido en su librería, la mano GIFU HAND III, utilizando varias de las funciones de modelado que incluye la herramienta e implementando el resto.

- Añadir la posibilidad de introducir y trabajar con objetos definidos en formato STL.

- Combinar funciones propias y funciones ofrecidas por la *toolbox* para calcular el agarre entre la mano añadida y un objeto stl con el fin de comprobar el correcto funcionamiento e implementación de estas ampliaciones.

Palabras clave: Mano robótica, objetos, formato STL, agarre.

Abstract

The main aim of this project is the understanding, utilization and improvement of a research toolbox, SynGrasp (in the development stage) designed to the mathematical software MATLAB.

This toolbox is used for calculating and analysing the grasp between different hands (humanoids or robotics) and objects.

To meet the aim, the work has focused on three key points:

- The addition to this toolbox of a hand model that is not contained in it, the GIFU HAND III, using modelling functions included in the toolbox and implementing some new functions.

- Adding the possibility of introducing and working with objects defined in STL format.

- Combine own functions and functions offered by the toolbox to calculate grasps between hand and a STL object in order to verify the correct operation and implementation of these extensions.

Keywords: Robotic Hand, objects, STL format, grasp.

Índice

Índice de Figuras.

Índice de Tablas.

1 Introducción.

A lo largo de la historia, el ser humano ha intentado mejorar sus condiciones de vida de diversas formas, entre estas formas podemos encontrar varias que se encargan de reducir de una forma u otra el trabajo realizado, ya sea porque este trabajo sea muy repetitivo o sea necesario el uso constante de esfuerzo físico para realizarlo. Gracias a esa ambición surgió y evolucionó una disciplina que hoy conocemos como robótica [1].

En la actualidad, la robótica se centra en crear artefactos cada vez más complejos que sean capaces de interaccionar con el mundo con mayor flexibilidad, y que sean capaces de reaccionar a cambios producidos en el mundo de forma autónoma. A esto hay que sumarle que los artefactos diseñados cada vez suelen tener un mayor parecido con partes del ser humano.

A finales de la década de los 50 el Dr. Engelberger diseño el primer robot industrial [2] que fue implantado en una cadena de montaje, la principal característica de este robot, fue su sistema de agarre en forma de pinza, por lo que fue el primer paso para llegar a las manos robóticas actuales.

Una mano robótica, es un componente que generalmente suele ser parte de otro llamado brazo robótico, que se puede diseñar para realizar cualquier tipo de tarea dependiendo de la finalidad del proceso [3]: como agarrar, soldar, o montar partes de una pieza.

Podemos encontrar dos tipos de manos robóticas; las manos utilizadas por autómatas en las cadenas de montaje (minimalista) o manos que intentan imitar la forma de la mano humana (antropomórfica) véase figura 1.

Figura 1. a) Robot Abb con mano robótica acoplada. b) Mano Gifu hand III sujetando una pelota.

1.1 Agarre de objetos con manos robóticas.

Como ya hemos visto, podemos diferenciar entre dos tipos de manos robóticas:

- Manos que intentan imitar la forma de la mano humana o **manos antropomórficas**. Éstas se utilizan para implantarlas en robot humanoides, para prótesis o para implementar nuevas tecnologías basadas en la interacción de la mano humana con el entorno e investigar y comprender sobre el agarre de las mismas.
- Manos utilizadas en cadenas de montaje cuyo objetivo es ser lo más eficientes o **manos minimalistas**. El principal objetivo de estas manos en la manipulación de objetos específicos, por lo que no es necesario que se parezcan a manos humanas, solo que realicen su función de forma eficiente y minimalista [4].

En los dos tipos de mano, el cómo agarrar objetos es un campo que es necesario analizar, pero las manos antropomórficas presentan un análisis más complejo por tener más grados de libertad (GDL) que el de las manos minimalistas.

La razón, es que el entorno humano esta creado y preparado para la interacción con las manos humanas, por lo tanto, las manos antropomórficas pueden agarrar lo mismo que agarra una mano humana, es decir, todo nuestro entorno puede presentar interacción con las manos antropomórficas porque ya está diseñado para interaccionar con la mano humana. También es por esta razón que el cálculo es más complejo, por la diversidad de formas y objetos que el entorno nos proporciona así como las distintas formas de agarrar un mismo objeto [5].

En el cálculo de agarre de objetos con manos robóticas se presentan varias dificultades que es necesario tratar. Para agarrar un objeto, cualquier tipo de mano tiene que tener un movimiento específico, planear la ruta que va a realizar para evitar obstaculizarse con partes del objeto que no son las destinadas a ser puntos de agarre, así como un sistema de control con sensores para evaluar la fuerza que se está ejerciendo sobre el objeto para evitar dañarlo y para confirmar el contacto con él. Pero el principal problema es que es necesario encontrar la forma de agarre que permita que el objeto no se caiga, pues puede estar la mano en contacto con el objeto pero no agarrándolo.

Uno de los principales problemas actuales en el cálculo de agarre, es la falta de precisión, ya que para agarrar objetos sencillos como una pelota los algoritmos son sencillos, pero para permitir que la mano recoja objetos más pequeños, o con geometrías más complejas [6] el factor precisión se debe tener en cuenta, así como plantear un algoritmo de acercamiento al objeto más complejo.

Estos problemas no afectan en gran medida a las manos minimalistas, pues los objetos con los que va a tratar en una cadena de montaje van a ser de la misma forma y sus movimientos van a ser muy limitados, ya sea elevar un objeto o atornillarlo, la forma del mismo no va a cambiar. Si la mano robótica minimalista presenta una forma de pinza y

puede tratar con diversos objetos tampoco es problema ya que el agarre no tiene que ser preciso solo debe ser válido para la función encomendada.

2 Manos robóticas.

2.1 La mano humana.

La mano, es la parte final de las extremidades superiores del cuerpo humano y es nuestra herramienta por excelencia.

Es un conjunto de huesos, músculos, tendones y terminaciones nerviosas [7] que nos permiten interactuar con el medio que nos rodea. Debido al número de terminaciones nerviosas que posee, se puede relacionar a la mano humana directamente con el sentido del tacto [5]. Hasta la fecha, la precisión y sensibilidad que se consigue con ella no se ha llegado a emular completamente por las manos robóticas [8].

La morfología de la mano se puede dividir en tres zonas:

- La muñeca o carpo, que es la parte de la mano que la une directamente con el brazo.
- La palma o metacarpo, de la que surgen los cinco dedos, ésta puede ser dividida en dos partes; la cara planar y la cara dorsal como se aprecia en la figura 2.
- Los dedos, representan el final de la mano y su parte más representativa son las falanges (huesos largos que los forman). Hay un total de cinco dedos; el pulgar, el índice, el medio, el anular y el meñique. Todos los dedos constan de tres falanges salvo el pulgar que consta de dos [7].

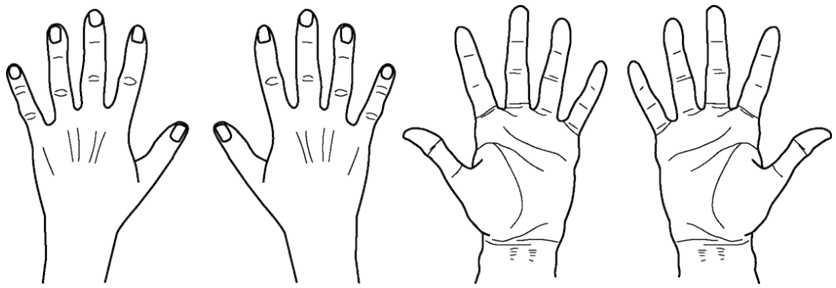

Figura 2. Cara dorsal y planar de la mano.

Uno de los factores más importantes de la mano, es el rango de movimiento de ésta y sus grados de libertad ya que limita qué cosas podemos alcanzar según qué posición tenga la mano.

Los **grados de libertad, GDL,** se refieren al movimiento que cada articulación puede realizar independientemente, es decir, la capacidad de moverse hacia delante/atrás, arriba/abajo, izquierda/derecha (traslación en tres ejes perpendiculares), combinados con la rotación sobre tres ejes perpendiculares (guiñada, cabeceo, alabeo) dando un total de 6 GDL como máximo por eje [9].

En una mano, como podemos ver en las figuras 3, 4 y 5, los movimientos de rotación cambian el nombre y son:

-Flexión / Extensión

-Abducción/Aducción
-Supinación/Pronación

Figura 3. Movimiento de extensión/flexión y aducción/abducción del pulgar.

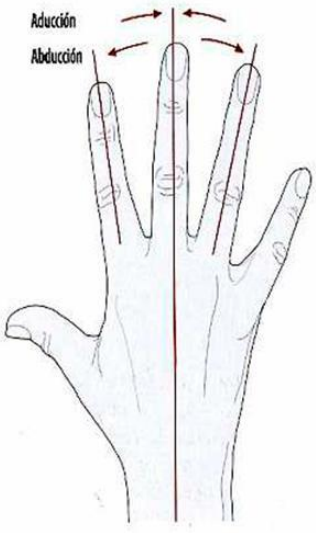

Figura 4. Movimiento de aducción/abducción del dedo medio.

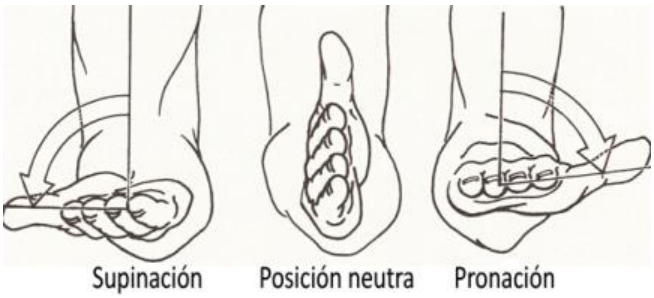

Figura 5. Movimiento de supinación/pronación de la mano.

De esta forma encontramos que en la muñeca hay 2 GDL: flexión/extensión y abducción/aducción [5].

Se puede añadir un tercer GDL si tenemos en cuenta los músculos del brazo, que es el que nos permite rotar la mano respecto al eje que forma el brazo produciendo supinación/pronación, ver figura 5.

El resto de GDL de la mano los aportan los dedos:

- El pulgar aporta un total de 5 GDL [10] como podemos ver en la figura 6: el primer GDL que encontramos es una pequeña flexión que se produce en la articulación trapezoescafoidea (TE) en la articulación trapeciometacarpiana (TM) y en la articulación metacarpofalángica (MF) encontramos 2 GDL flexión/extensión y abducción/aducción, finalmente un GDL de flexion/extensión en la articulación interfalángica (IF).

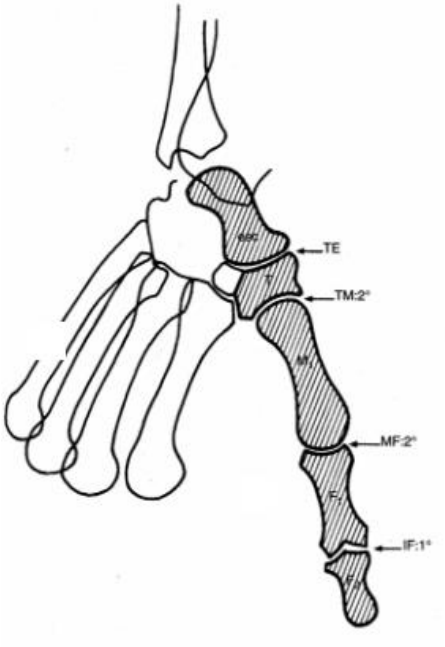

Figura 6. Articulaciones del pulgar.

- Cada dedo restante aporta un total de 4 GDL, en la primera articulación de cada dedo se presentan 2 GDL, abducción/aducción y flexión/extensión, y en las dos articulaciones restantes encontramos flexión/extensión, como podemos ver en la figura 7.

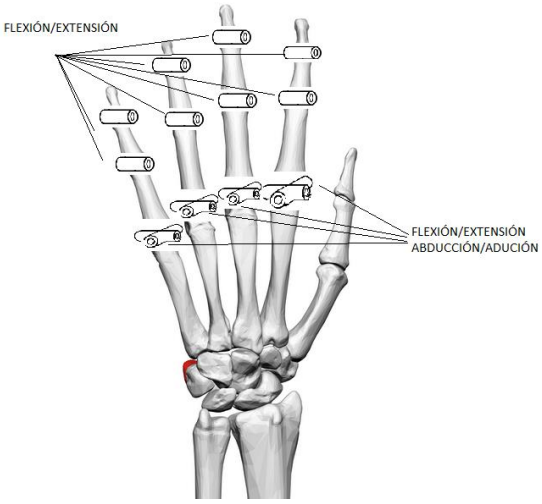

Figura 7. GDL de los dedos índice anular medio y meñique.

Por lo tanto, la mano humana tiene un total de 24 GDL, lo que permite situar la mano de muchas formas para poder coger un objeto o realizar alguna acción.

Por otro lado, los arcos de movilidad determinan los límites que puede recorrer una mano en su movimiento sin llegar a lesionarse, y por tanto limitan la posición que tiene que tener una mano para coger un objeto [7].

Este último aspecto es el que presenta más ambigüedad, pues según distintos autores los límites de movilidad pueden variar [11].

Partiendo del reposo (mano alineada con el eje del brazo) la muñeca puede flexionarse 80º y extenderse 70º, además permite una aducción de 30º y una abducción de 20º [12], como apreciamos en la figura 8.

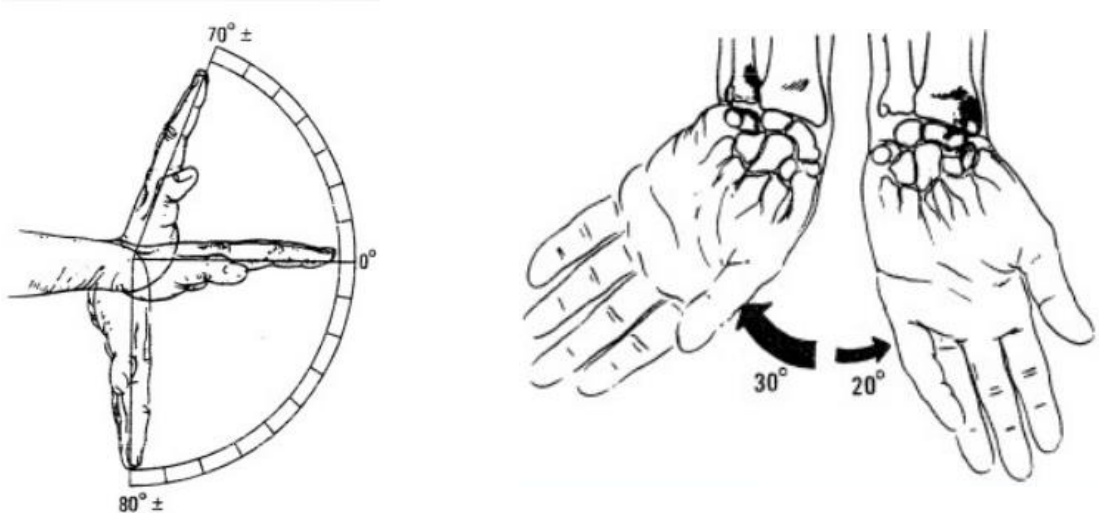

Figura 8. a) Límites flexión/extensión de la muñeca. b) Límites aducción/abducción de la muñeca

Los límites para las articulaciones de los dedos, salvo el pulgar, como vemos en la figura 9, son los siguientes [7]:

- Para la primera articulación, una flexión de 90º y una extensión de 30-40º.
- La segunda articulación presenta una flexión de 110º y una extensión de 0º.
- La última articulación presenta 90 º de flexión y 10º de extensión.
- Los límites de abducción entre cada dedo son unos 20º y de aducción los dedos llegan a tocarse unos con otros.

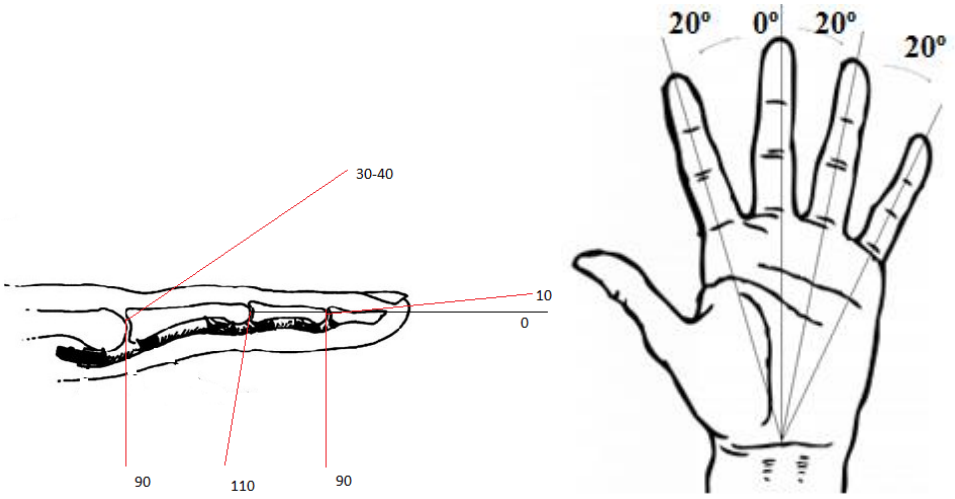

Figura 9. a) Límites de flexión extensión de un dedo. b) Límites de abducción de los dedos.

Por último, el análisis del pulgar presenta más dificultad al tener la capacidad de anteponerse a la palma [13] y por tanto presenta nuevos tipos de movimiento:

Su articulación MF, ver figura 8, presenta flexión de 90º y extensión de 30º, mediante la aducción puede llegar a tocar al dedo contiguo, mientras que con la abducción puede alejarse hasta 60º. La articulación IF puede flexionarse hasta 45-80º y oponerse 120º. Finalmente si la palma se coloca horizontalmente el pulgar puede abducirse hasta 70º [7], como se aprecia en la figura 10.

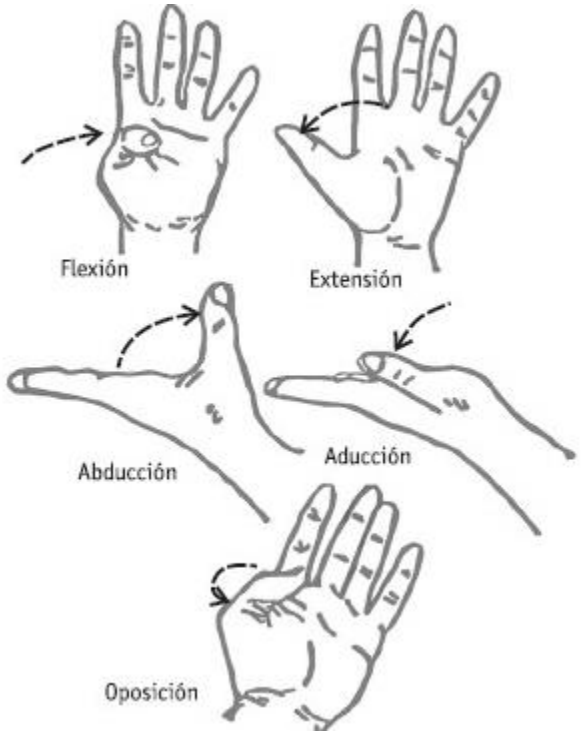

Figura 10.Movimientos del pulgar.

Como hemos visto, la mano humana es muy compleja, y a diferencia del resto de seres vivos, cuenta con un pulgar oponible muy desarrollado. Siendo este el factor más importante para la evolución humana ya que gracias al pulgar, la mano humana cuenta con el agarre de pinza, el cual permitió desarrollar herramientas e interactuar con el entorno, siendo esta característica la que nos llevó a desarrollar lo que hoy conocemos como tecnología.

Por todo esto, se está trabajando actualmente en desarrollar manos robóticas antropomórficas, por la versatilidad que poseen, por la cantidad de formas de poder coger objetos y por ser herramientas capaces de fabricar e interactuar con todo lo que el ser humano ha sido capaz de construir, pues todo lo que el ser humano ha desarrollado está pensado para ser usado por él, y por tanto, también está pensado para manos robóticas antropomórficas, abriendo un mundo de posibilidades para éstas.

2.2 Diseño de una mano antropomórfica.

Como hemos visto, la mano humana es muy compleja y presenta un gran número de músculos, tendones y huesos que funcionan en sincronía para lograr que esta se mueva.

Por lo que a la hora de intentar plasmar o imitar la funcionalidad de la mano, es necesario seguir unas pautas de diseño, y modificar algunas características para solucionar los problemas que puedan aparecer. Como puede ser: calcular la cinemática de cada dedo, la construcción e integración de los actuadores que formen la mano o la implementación del sistema de control [14].

Los primeros conceptos básicos que se deben tener en cuenta en el diseño es el de una mano antropomórfica son los de cinemática, la superficie de contacto y el tamaño:

- La cinemática de una mano antropomórfica debe coincidir con la de la mano humana, es decir, debe presentar un pulgar oponible una palma y dedos con varios GDL.
- La superficie de contacto de la mano antropomórfica debe ser lisa y uniforme (zona de contacto libre de cables u otros obstáculos para evitar problemas) y debe coincidir o reflejar los aspectos de una mano humana, es decir, coger un objeto como una mano humana y no presentar incompatibilidad como por ejemplo, a tornillos que unan dos partes de la mano.
- El tamaño, si una mano robótica pretende parecerse a una humana el tamaño entre ambas debe ser similar.

Actualmente, además de estas características en el diseño, se está intentando introducir un concepto de diseño nuevo, diseño en arcos flexibles, que aparece en la figura 11. Ya que gracias a la capacidad de juntarse y extenderse de estos arcos, se aumenta el control sobre los objetos agarrados y permite a la mano antropomórfica coger objetos de mayor tamaño o de geometría más compleja [15] [16].

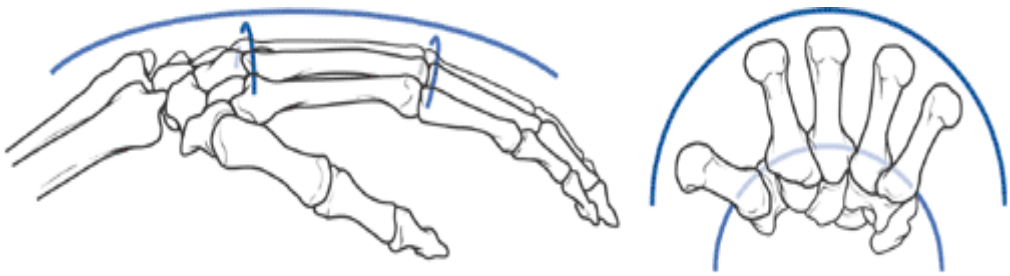

Figura 11. Arcos de la mano humana.

Una vez analizados los primeros requisitos que debe cumplir una mano antropomórfica, aparecen problemas de diseño [14]. Por ejemplo, que el tamaño de los componentes necesarios para mover una mano robótica puede ser demasiado grande para el tamaño de la mano. Para solucionar este problema se puede seguir dos caminos:

- Aumentar ligeramente el tamaño de las partes de la mano para colocar los actuadores en las partes que les corresponden.
  Estos actuadores, pueden ser de dos tipos: actuadores de conducción directa, si se colocan en la articulación directamente, o actuadores *link hosted actuation (*el actuador se sitúa en el eslabón entre dos articulaciones y ésta se mueve gracias a una cadena cinemática desde el actuador).
  Generalmente esta solución se utiliza para disminuir la complejidad de la trasmisión y se suele implementar en articulaciones independientes de otras.

- Otra posible solución, es colocar los actuadores fuera de los eslabones que van a manejar. Esta solución, es utilizada cuando el tamaño es muy limitado (pues acarrea más problemas que la solución anterior). Por ejemplo, si se sitúan los motores de los dedos demasiado alejados de éstos, es necesario implementar una cadena de movimiento más compleja.

2.3 Partes de una mano antropomórfica.

Una vez claros los conceptos de diseño, así como los problemas que pueden aparecer y como solucionarlos, hay que tener en cuenta cuáles son las principales partes que tiene una mano antropomórfica [14]:

Estructura.

Es el cuerpo de metal u otro material formado por los eslabones y articulaciones que da forma a la mano.

Actuadores.
Un actuador, es un dispositivo mecánico, cuya función es actuar sobre un elemento o proporcionarle una fuerza utilizando energía para ello, generalmente, energía eléctrica, neumática o hidráulica [17].
Según el número de articulaciones que posea la mano robótica, hay un número de actuadores, pero en una mano antropomórfica, puede haber articulaciones que no se muevan o puede haber varios actuadores que actúen sobre una misma articulación o sobre varias [18] (utilizando distintas arquitecturas como puede ser la de barras), es decir, hay varios tipos de manos, y por tanto, hay varios tipos de arquitecturas de actuadores para cada mano que se pueden agrupar dentro de dos grupos [14]:

- *Actuadores de simple efecto*: los motores, sólo controlan el movimiento en una dirección, por lo que el movimiento contrario tiene que venir dado por una fuerza

externa, como puede ser otro motor o un elemento pasivo. Se pueden dividir según con qué elemento funcione el actuador en varias arquitecturas:

- Actuadores simples con retorno pasivo: estos actuadores utilizan la fuerza absorbida por el elemento pasivo, como puede ser un muelle, mientras el actuador está funcionando para devolverlo a su estado de reposo cuando este deja de ejercer fuerza. Podemos ver su esquema en la figura 12.

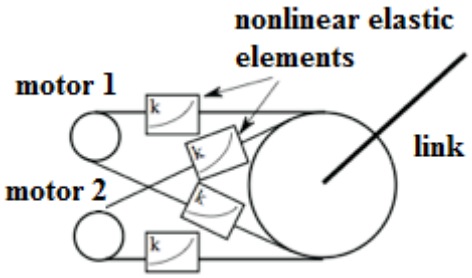

Figura 12. Actuador con retorno pasivo.

- Dos actuadores de simple efecto: un actuador para cada movimiento de la articulación. Este método, aporta varias ventajas respecto al anterior, ya que al poder controlar dos actuadores a la vez, se puede generar un par de accionamiento y modificar la rigidez del eslabón que se controla (como puede ser un dedo de la mano) aportando más verosimilitud con una mano humana, el problema de éste, es el tamaño, ya que es más grande [19]. Podemos ver su arquitectura en la figura 13.

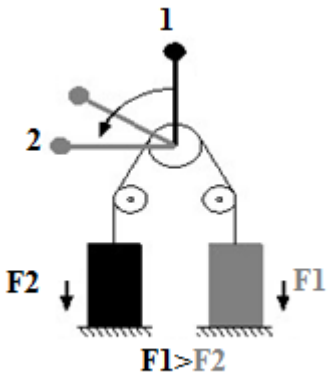

Figura 13. Dos actuadores de simple efecto.

- Actuadores de simple efecto organizados en red: esta arquitectura, imita la configuración de un sistema biológico, consiste en tener un número de actuadores comprendido entre el número total de articulaciones y el doble de ese número. De esta forma, el movimiento de cada articulación estará gobernado como mínimo por dos actuadores (resultando el caso anterior). La utilidad de este tipo de actuadores, es controlar la rigidez según el

agarre, pero además permite modificar las velocidades de la articulación a lo largo de un recorrido reduciendo problemas de fricción. Al igual que la arquitectura anterior, el principal problema es el tamaño, añadiéndole además la complejidad del diseño y del control de este tipo de actuadores.

- *Actuadores de doble efecto*: en este caso, los motores controlan el movimiento en ambas direcciones. Este tipo de actuadores está pensado para usarse cuando hay esfuerzos elevados de torsión en ambas direcciones [20].

Otros tipos de actuadores

Hay otros tipos de actuadores que pueden tener las manos robóticas, como la *Shadow dexterous hand*, que posee lo que se conoce como músculos neumáticos, que es un tipo de actuador neumático desarrollado por *Festo*.

Está formado por un cilindro hueco de elastómero con fibras de aramida. Cuando el músculo neumático se llena de aire, su diámetro aumenta y su longitud se contrae, esto permite un movimiento fluido-elástico.
La fuerza que produce este músculo neumático es superior (10 veces) la fuerza que puede realizar un cilindro neumático de su tamaño, incluyendo que gracias a este actuador se pueden imitar los movimientos que realizan los músculos humanos respecto a su cinética, velocidad, fuerza e incluso precisión. [21, 22]. Podemos ver este actuador en las figuras 14 y 15.

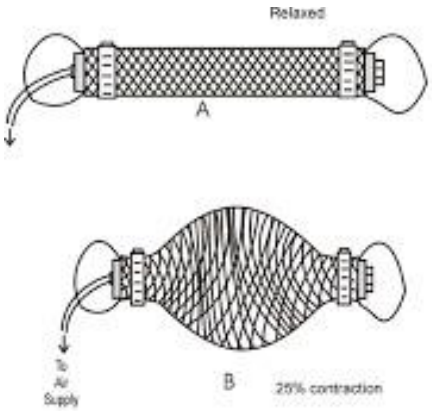

Figura 14. a) Músculo neumático relajado
b) músculo neumático con un 25% de contracción.

Figura 15. Mano robótica con el sistema de músculos neumáticos Shadow dexterous.

Pese a ello, este actuador también presenta una serie de problemas para su implementación, entre ellos los más destacables son que requiere una bomba neumática para su funcionamiento, o la necesidad de un sistema de control que pueda hacer frente

a un retraso entre la señal de control de movimiento y la acción muscular eficaz (debido a que el gas es compresible) [23].

Sensores táctiles

Un sensor, es un dispositivo capaz de recibir o detectar cambios o estímulos del exterior. En este caso, un sensor táctil, es capaz de detectar si el sensor está en contacto con algún elemento externo. Este tipo de sensores no solo pueden detectar si un objeto está en contacto con él, también es capaz de medir la fuerza que se produce entre ellos, fuerza de contacto [24].

Los sensores táctiles pueden ser de varios tipos:

- Sensores de fuerza, útiles en una mano robótica por su baja histéresis (conservar su estado una vez desaparecida la fuerza) y por su bajo rozamiento interno.
  Estos sensores, en las manos robóticas miden el par de torsión que se está generando en la articulación y su cálculo es sencillo (si los motores son de corriente continua), ya que es proporcional a la corriente utilizada por el actuador que está ejerciendo fuerza sobre esa articulación.
- Sensores de matriz táctil, como el que vemos en la figura 16, estos sensores son una matriz de sensores de fuerza y permiten además de calcular la fuerza, reconocer la forma del objeto con el que se está interactuando.

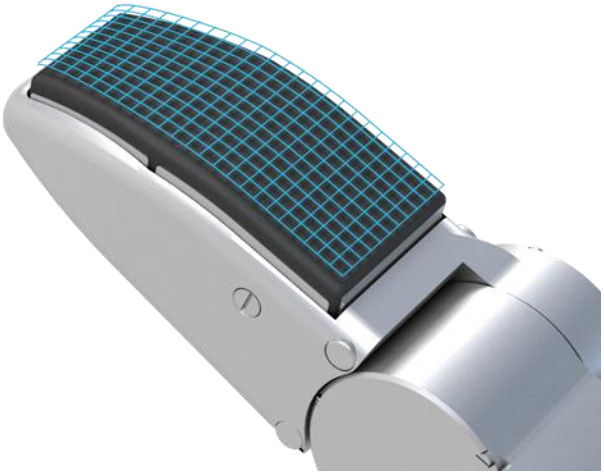

Figura 16. Matriz de sensores táctiles en un dedo de la mano
SDH servo-electric 3-finger gripping hand.

- Sensores capacitivos. Detecta los cambios de capacitancia que se producen por el acercamiento de objetos a él.

Sistema de control.

El sistema de control, es la parte de la mano robótica que recibe la información de los sensores, la analiza y la trata para dar una respuesta, según los resultados obtenidos a los actuadores de la mano.

El sistema de control en una mano robótica antropomórfica, puede tratarse como si fuese un conjunto de manipuladores tradicionales, pero con varios problemas asociados a la no linealidad que estos tienen, originada por la fricción, las lecturas no legibles de los efectos dinámicos, el *blacklash*, las complicaciones en el sistema de transmisión de información, la mala colocación de sensores y actuadores por no tenerse en cuenta en la fase de diseño y el tipo de sistema según qué actuadores se hayan utilizado [14].

2.4 Ejemplos de manos robóticas.

Como ya hemos visto, hay varios componentes y arquitecturas con los que diseñar una mano robótica y su número es cada vez más elevado, Por eso, es necesario citar algún ejemplo de éstas, para tener una clara visión de ellas y ver sus variedades, como pueden ser el distinto número de GDL [25, 26]:

Shadow Dexterous Hand [27]

Una de las manos antropomórficas más conocidas, es la *Shadow dexterous Hand*, de la compañía Shadow que, como vimos en el apartado anterior, tiene la peculiaridad de la utilización de músculos neumáticos como actuadores en algunos modelos, como el de la figura 17. Es una mano que ha tenido varios prototipos, como pueden ser el C5M o el C6M o la serie E1, con 5 dedos, el pulgar posee 5 GDL y el resto de dedos 3 GDL, y es de las primeras manos antropomórficas que se comercializaron [5].

Figura 17. Shadow Dexterous Hand.

EH1 Milano Hand [28]

Perteneciente a la compañía Prensilia, la mano antropomórfica EH1 Milano, que podemos ver en la figura 18, posee un total de 5 dedos que suman 16 GDL y 6 motores que controlan el movimiento de la mano. La versión anterior de esta mano, fue la primera controlada por la mente de un voluntario como una prótesis, mediante unos electrodos colocados en su extremidad, en el año 2008.

Figura 18. EH1 Milano Hand.

Actuated Sheffield Hand [29]

Mano antropomórfica de la compañía Elumotion basada en una réplica de la mano de la Universidad de Sheffield Orebro con un total de 5 dedos y 12 GDL, sus componentes de actuación fueron diseñados en módulos capaces de realizar la flexión y extensión de cada dedo. La peculiaridad de esta mano es la utilización de varillas telescópicas que imitan los tendones de los dedos de una mano humana, estas varillas pueden apreciarse en la figura 19.

Figura 19. Prototipo diseñado por la Universidad Orebro, Sheffield Hand.

Elu 2 Hand [30, 31]

La mano Elu2, como podemos ver en la figura 20, es una mano a escala real respecto a la humana. Es capaz de aproximar movimientos reales de las manos humanas a velocidades similares, tiene un total de 5 dedos y 9 GDL que son accionados dentro de la propia mano, gracias a esto la mano puede ser montada en distintos tipos de brazos, pues los motores no se encuentran en éstos.

Figura 20. Elu 2 Hand agarrando una pelota.

Robonaut Hand [5] [32]

Mano implementada por la NASA con 5 dedos, 12 GDL en la mano y 2 GDL en la muñeca, 14 motores que mueven las articulaciones y 43 sensores. Esta mano se implementó para el robot antropomórfico Robonaut que fue pensado para realizar tareas en el espacio. Esta mano la podemos encontrar en la figura 21.

Figura 21. NASA Robonaut Hand.

SR finger [33]

Desarrollado por el MIT, es una mano robótica no antropomórfica, que solo consiste en 2 dedos, como se aprecia en la figura 22, que actúan en forma de pinza, está pensado para funcionar como una extensión de la mano humana para convertirla en una mano de 7 dedos y facilitar tareas, permitiendo realizar tareas en las que se necesitan dos manos con solo una.

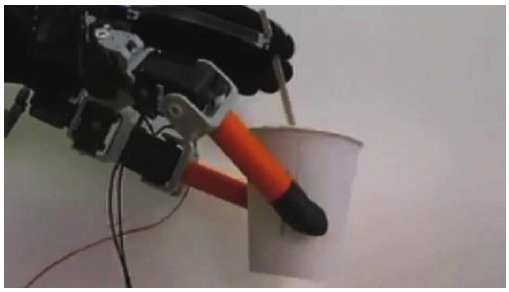

Figura 22. SR finger sujetando una taza de café.

MA-I [34]

La mano antropomórfica MA-I (Mano Artificial Inteligente), figura 23, está diseñada y construida en la Universidad Politécnica de Cataluña, tiene un total de 4 dedos, con 4 GDL cada uno, dando un total de 16 GDL.

Figura 23. Mano MA-I.

DLR Hand II [35]

La mano antropomórfica DLR Hand II, mostrada en la figura 24, fue diseñada por el centro de robótica y mecatrónica de la DLR (agencia aeroespacial alemana). Ésta consta de 4 dedos idénticos en tamaño, con 4 ejes y 3 GDL cada uno. El número total de GDL es 13 ya que en la palma tiene un GDL extra.

Figura 24. DLR Hand II.

Barret Hand [36, 37]

Es una mano robótica no antropomórfica, de 3 dedos y con un total de 7 GDL, 2 por cada dedo y uno en la muñeca, suele ir enganchada en el brazo robótico Barret, como vemos en la figura 25, y es capaz de agarrar objetos de distintas formas y tamaños.

Figura 25. Barret Hand.

3 Gifu Hand III.

De todas las manos antropomórficas que podemos encontrar, para este proyecto se ha utilizado la mano Gifu Hand III.

El principal motivo de esta elección es que el departamento de sistemas y automática de la universidad posee una Gifu Hand III (es la primera universidad europea que ha adquirido la mano para proyectos de investigación [38]) por lo que tener un programa de simulación de agarre y un estudio detallado de la misma, es bastante beneficioso para futuros proyectos a realizar con la mano en el departamento.

La mano Gifu Hand III fue diseñada y construida por la universidad de Gifu de Japón. Esta universidad realiza proyectos e investigaciones sobre robótica en su departamento *Kawasaky And Moury Laboratory*, que tiene como objetivo estudiar y analizar la inteligencia de los sistemas mecánicos y la interfaz humana con la intención de realizar autómatas tan hábiles como los humanos para realizar su trabajo [39].

Uno de los proyectos de esta universidad dirigido por Kawasaki, Komatsu y Mouri se inició en la década de los 90 [40] y consiste en la fabricación de manos multi-dedo para robots [41], [42], [5].

Este proyecto, hasta la fecha, ha diseñado un total de 4 manos antropomórficas, cada una como mejora de la mano anterior, siendo estas manos: Gifu Hand I, la primera de todas ellas fabricada en el año 1997 [41], la segunda mano creada a partir de esta fue la Gifu Hand II en el año 1999 (a partir de ella se creó la mano que hemos simulado), la Gifu Hand III del 2002 y por último a partir de la Gifu Hand III se ha creado la mano KH Hand Type S en el año 2005 [5].

Como todas las manos proceden de mejoras hechas de la primera, todas son muy similares en muchos aspectos, como pueden ser el tamaño y la forma. Estos tipos de mano fueron diseñados para realizar tareas de agarre, manipulación y presentar una forma antropomórfica siguiendo estos conceptos de diseño (muy similares a los que vimos en el apartado 2.2) [41]:

- Tamaño: semejante al de la mano humana para conseguir una manipulación hábil, poseen un bajo peso.
- GDL: el número de ejes y el número de GDL de cada dedo debe ser similar a los dedos de una mano humana.
- Pulgar oponible: el pulgar de estas manos robóticas puede oponerse al resto de los dedos permitiéndolas manipular los objetos con la destreza propia de una mano humana.
- Sensores de Fuerza: el agarre con una mano robótica se consigue gracias al uso de sensores táctiles y de fuerza en los dedos de ésta.

- Servomotores incluidos en la propia mano: construir un diseño en el que los motores estén incluidos en la mano para evitar problemas o fallos si se acopla la mano a un brazo robótico.
- Dedos estructurados: cada conjunto debe ser modular para facilitar la limpieza y el mantenimiento de la mano.

Pese a sus similitudes aparecen varias diferencias propias de la evolución tecnológica y de diseño, como pueden ser, el cambio del tipo de material utilizado para su fabricación, mejoras mecánicas, variación en el número y tipo de sensores, variación del tipo de motores de la mano o incremento de estos.

Para ver las principales partes de la mano, se hará una descripción de las 4 manos del proyecto Gifu:

Gifu Hand I [43, 44]

La mano robótica Gifu Hand I, que podemos observar en la figura 26, está compuesta de una palma, un pulgar y 4 dedos. Su estructura consta de 20 articulaciones y un total de 16 GDL con un sistema de servomotores *built-in* (motores en la propia mano) y un sistema de sensores a lo largo de toda la mano. Sus servomotores son *Maxson DC* fabricados por *Interlectric AG*, cada servomotor consta de un encoder magnético de 16 cuentas/revolución para medir el ángulo de giro de éstos. Es muy ligera y compacta, 1226 gf (gramo fuerza, Sistema Técnico de Unidades 1 Newton = 101,97 gf) dando aproximadamente un total de 1,22 Kg.

El pulgar presenta 4 articulaciones y 4 GDL, mientras que el resto de los dedos 3GDL. El pulgar tiene otra diferencia respecto al resto de los dedos, mientras que la estructura de servomotores para mover la última articulación del resto de dedos consiste en un servomotor conectado con cuatro varillas planas, en el pulgar se mueve por un servomotor y un tren de engranajes situado en su segundo eslabón.

Figura 26. Gifu Hand I.

Gifu Hand II [45]

La mano robótica Gifu Hand II, que encontramos en la figura 27, mantiene la mayoría de componentes y diseño de la mano Gifu Hand I. Esta mano fue diseñada teniendo en cuenta una serie de mejoras críticas en el funcionamiento respecto a la anterior. Las principales mejoras son:

- Reducción del *Blacklash* (lecturas no legibles provocadas por los efectos dinámicos de la mano): la Gifu I incluía un mecanismo de reducción diferencial asimétrico formado por engranajes cónicos entre el 1º y 2º eslabón de cada dedo. Este mecanismo permitía situar la segunda articulación de cada dedo cerca de la palma. El problema que aparecía era que los engranajes cónicos generaban una fuerza axial que desplazaba el segundo engranaje provocando blacklash y generaba abrasión en el material de la Gifu I (aluminio) incrementando aún más el blacklash. La solución implementada en la Gifu II fue sustituir los engranajes cónicos por frontales y cambiar el material de la mano a titanio, reduciendo el error de blacklash de la primera articulación de 8 grados a 1 y en el resto de articulaciones una cantidad parecida.
- Mayor par de salida y mayor respuesta: Para lograrlo usaron motores de más potencia y un mayor ratio de reducción en la 1º y 2º articulación de cada dedo y añadieron a la retroalimentación del sistema de control un nuevo PD. Esta mejora además hizo posible ignorar el efecto de la gravedad en los movimientos de los dedos y eliminar el problema de la vibración provocando que la mano Gifu II se pueda mover más rápido.
- Mayor rigidez: utilizando un análisis de elementos finitos se consiguió aumentar la rigidez de la mano.

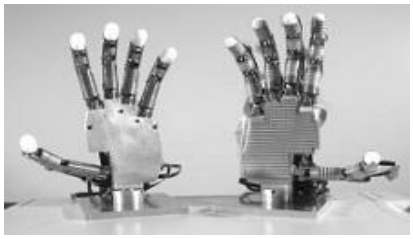

Figura 27. Gifu Hand II.

Gifu Hand III [46]

La mano elegida para este proyecto, Gifu Hand III, que podemos encontrar en la figura 28, consta de un pulgar y 4 dedos, el pulgar lo forman 4 articulaciones y proporciona 4 GDL y el resto de dedos lo forman 4 articulaciones pero proporcionan solo 3 GDL cada uno, dando un total de 20 articulaciones y 16 GDL.

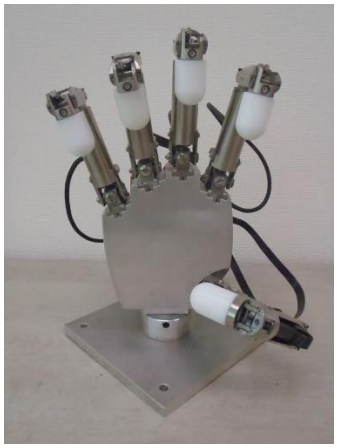

Figura 28. Gifu Hand III.

Ya que esta mano es la mano elegida, podemos ver en la figura 29, un análisis de las dimensiones y estructura de la mano completa, Así como un análisis más detallado del diseño y mecanismos de los dedos y pulgar de ésta en la figura 30.

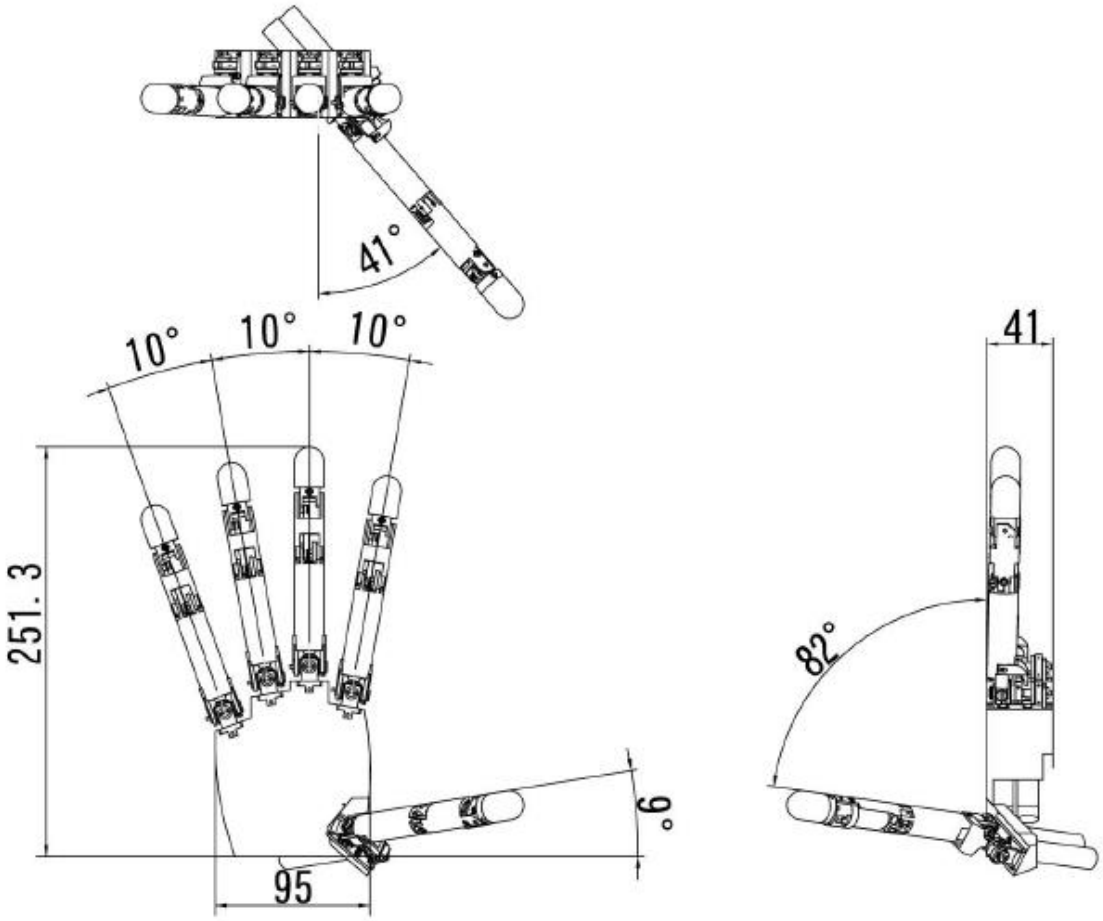

Figura 29. Estructura y dimensiones de la mano Gifu III.

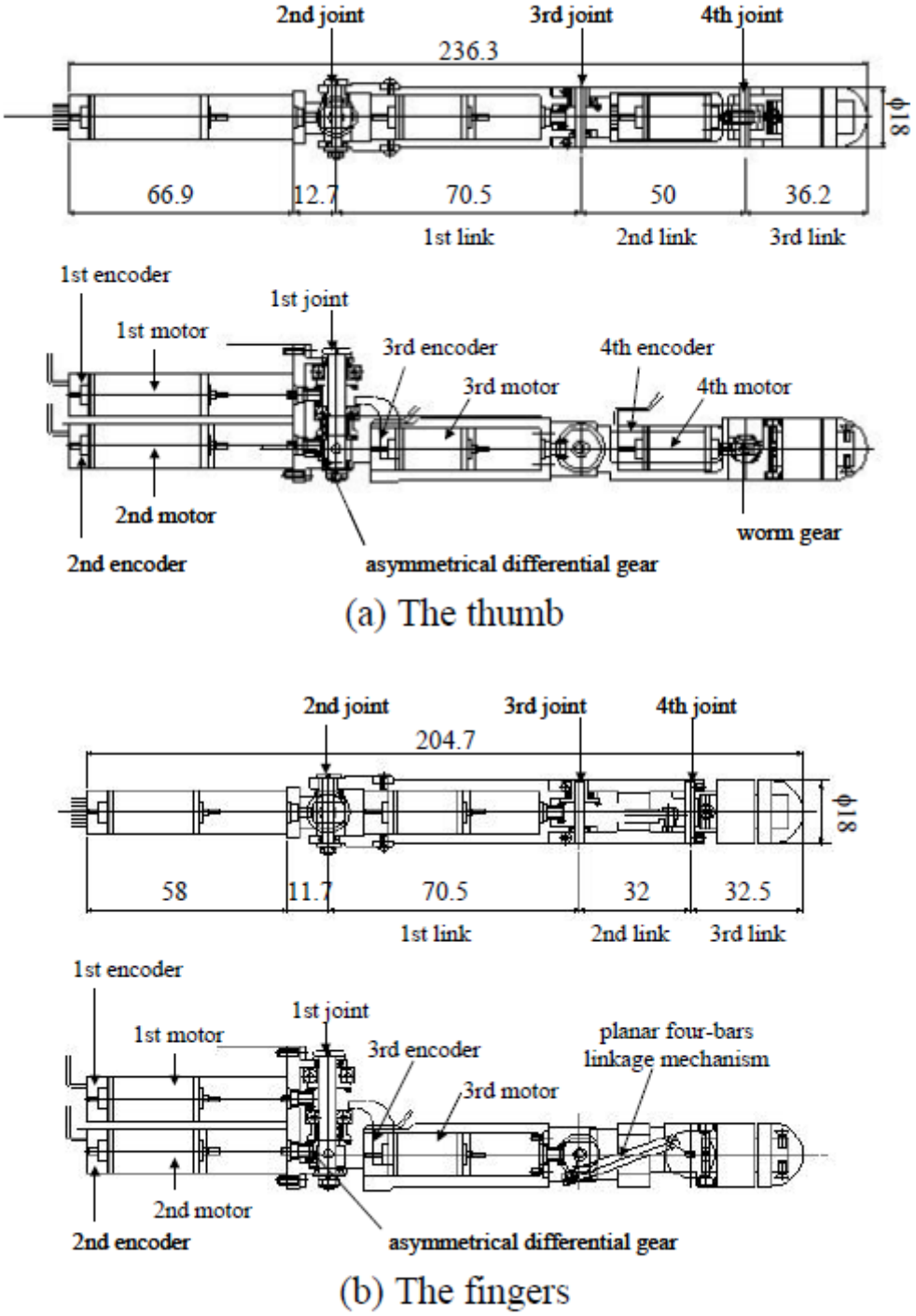

Figura 30. a) Dimensiones y mecanismo del pulgar de la mano Gifu III.
b) Dimensiones y mecanismo de los dedos de la mano Gifu III.

Cada servomotor de la mano tiene un encoder magnético de 16 pulsos/rev, la tabla 1 muestra las especificaciones de la mano.

Weight	Thumb	0.25 [kgf]
	Finger	0.20 [kgf]
	Total	1.4 [kgf]
Operating angle of joints	1st joint	-28 ~ 28 [deg] (Thumb) -20 ~ 20 [deg] (Finger)
	2nd joint	-10 ~ 90 [deg]
	3rd joint	-10 ~ 90 [deg]
	4th joint	-10 ~ 90 [deg]
Output force at fingertip	Thumb	3.7 [N] [1], 2.8 [N] [2]
	Finger	3.4 [N] [1], 1.8 [N] [2]
Output torque of the thumb	1st joint	1.76 [Nm] [1], 1.03 [Nm] [2]
	2nd joint	1.27 [Nm] [1], 0.58 [Nm] [2]
	3rd joint	0.33 [Nm] [1], 0.25 [Nm] [2]
	4th joint	0.04 [Nm] [1], 0.02 [Nm] [2]
Gear ratio of the thumb	1st joint	713.43:1
	2nd joint	384.69:1
	3rd joint	148.48:1
	4th joint	80.00:1
Band width of the thumb	1st joint	10.4 [Hz]
	2nd joint	12.3 [Hz]
	3rd joint	7.4 [Hz]
	4th joint	9.5 [Hz]

[1]Maximum output, [2]Rated output

Tabla 1. Especificaciones de la mano Gifu Hand III.

La mano Gifu III presenta una serie de mejoras respecto a la mano Gifu II, las principales son:

- Reducción del *Blacklash*: pese a que la Gifu II ya había mejorado este aspecto y el error de *blacklash* estaba cerca de 1 grado, después de operaciones largas este error se incrementaba hasta más de 5 grados. La solución implementada en la Gifu III fue colocar los engranajes de forma simétrica a lo largo del eje de la primera articulación y añadir una correa para evitar el desgaste de estos consiguiendo reducir el *blaklash* a menos de 1 grado.
- Mayor respuesta: una mayor respuesta en la primera articulación del pulgar añadiéndole un sistema de control PD consiguiendo una frecuencia de ancho de banda de 10,4 Hz. En el resto de dedos también se implanta esta mejora siendo la

mínima frecuencia obtenida en la mano de 7,4 Hz. Como el ancho de banda de un dedo humano es 5,5 Hz esta mano logra moverse más rápido que la mano humana.

- Mejora de la oposición del pulgar: para conseguir esta mejora los dedos de la Gifu III se han recolocado respecto a los de la Gifu II, gracias a esto se ha incrementado el área de operación de la mano al que se muestra en la figura 31.

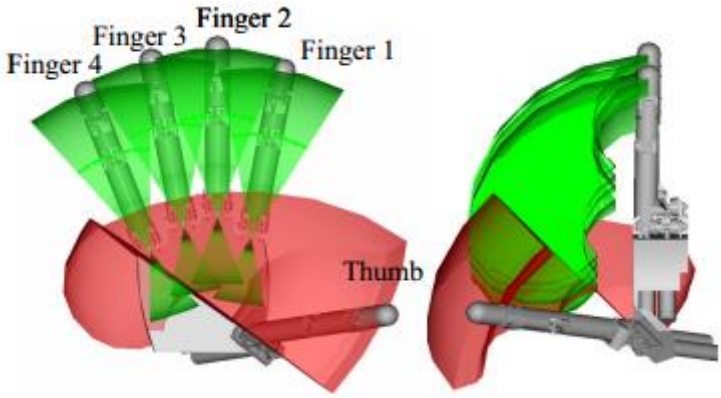

Figura 31. Área de operación.

- Amplificador de potencia de 16 canales: utilizado para supervisar las corrientes eléctricas, ya que la corriente eléctrica cambia en función del par del motor. Del análisis de esta relación se puede obtener el valor de la fricción en las articulaciones de la mano.
- Mejora del mecanismo de 4 barras planas: el ángulo de giro de la 3º articulación de cada dedo modifica el ángulo de la 4º articulación de una forma más parecida a la de la mano humana.

KH Hand Type S [41]

La última de las manos KH Hand Type S, de la figura 32, se puede considerar una versión de la mano Gifu Hand III por lo que no presenta apenas cambios respecto a ésta. Sus principales mejoras son:

- Reducción del *Blacklash*: El *blacklash* de esta mano se reduce aún más que en la anterior.
- Reducción del peso: la mano KH Hand Type S consigue reducir su peso a 1,09 Kg respecto a los 1,4 Kg que pesaba la Gifu III utilizando plástico como material principal en lugar de titanio.
- Cambio de los motores y los encoders magnéticos: Con el fin de reducir el peso esta mano utiliza motores más pequeños y cambia sus encoders magnéticos de 16 pulsos/rev a 12 pulsos/rev. Esto proporciona un aumento de la velocidad de los dedos pero a cambio reduce su fuerza 0,48 veces respecto a la Gifu III.
- Nuevo sistema de cableado: El sistema de cableado en estas manos se coloca por la parte posterior de la mano, aun así los accidentes en tareas de agarre podían ocurrir si la forma del objeto era compleja o presentaba cavidades, para evitarlo

en esta mano el sistema de cableado es mucho más compacto como podemos ver en la figura 33.

- Aumento del número de sensores táctiles: un leve aumento en la cantidad de puntos del sensor (36) dando un total de 895.
- Aumento de la velocidad: con este aumento la mano es capaz de realizar el lenguaje de signos de forma fluida.

Figura 32. KH Hand Type S.

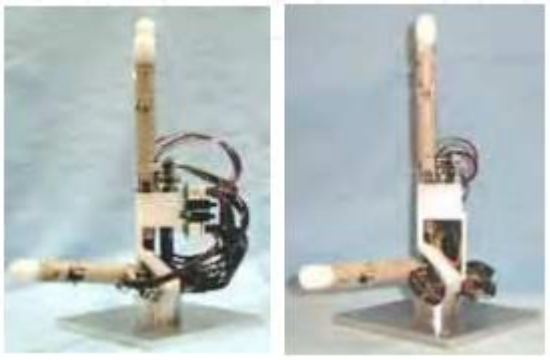

Figura 33. Antes y después del sistema de cableado.

4 Estado del arte de los simuladores de agarre.

Debido a la cantidad de proyectos y estudios que se pueden realizar con manos robóticas, el número de ellas ha ido aumentando y los simuladores de robótica han ido incluyendo ciertas mejoras o herramientas para trabajar con ellas, y así, realizar los cálculos necesarios para moverlas o combinarlas con otra serie de elementos robóticos.

Entre estas mejoras o herramientas nuevas, encontramos los simuladores de agarre (que es la función principal las manos) que han surgido para optimizar y realizar los cálculos necesarios para mover estas manos y conseguir agarrar objetos de una forma óptima y segura.

Otra ventaja que proporciona este tipo de simuladores, es la posibilidad de poder utilizar y simular agarres de objetos sin necesidad de tener manos robóticas físicamente (utilizando para esto los bancos de prueba que incluyen), o para implementar los resultados después de ser evaluados en los bancos de prueba y así conseguir evitar dañar la mano robótica.

Podemos encontrar varios simuladores de agarre gratuitos, los más conocidos son:

GraspIt! [47]

El simulador GraspIt! , cuya interfaz encontramos en la figura 34, fue desarrollado por *Robotics Lab* del departamento de informática de la universidad de Columbia (2002). Es un simulador que puede colocar arbitrariamente manos robóticas y cargar objetos y obstáculos (también arbitrariamente), consiguiendo con esto, simular completamente un mundo.

GraspIt! incluye un sistema de detección de colisión y un sistema para determinar puntos de contacto entre dos cuerpos, permitiendo así, simular una interacción entre manos y objetos. Además de proporcionar los resultados de estas simulaciones, GraspIt! permite evaluar la calidad del agarre entre mano y objeto y mostrar una representación gráfica de la posición final de la mano con el objeto agarrado, así como los puntos más débiles que aparecen en el agarre.

El concepto de funcionamiento para agarrar objetos de este simulador es sencillo: Se coloca la mano robótica en una posición, de tal forma que el objeto a coger esté dentro de su alcance, después se realizan varios intentos de agarre cerrando la mano para finalmente mostrar el mejor de todos.

La metodología que incluye en el análisis del agarre es la siguiente:

- El sistema de detección de colisión funciona de forma que si una colisión es detectada, los cuerpos retroceden a su posición anterior y el programa guarda la distancia que había en ese momento (GraspIt! considera contacto si la distancia entre los dos cuerpos es menor a 0,01mm) entre los dos puntos.

- Genera una zona de contacto entre los dos cuerpos (los cuerpos están definidos por conjuntos de triángulos), cualquier par de triángulos de puntos (uno de cada objeto) que tengan una distancia igual o menor a la guardada en el paso anterior, será considerado punto de contacto.
- Con los puntos que obtiene el simulador calcula la fuerza y la fricción que se ejerce usando para ello el modelo de Coulomb.
- Por último, GraspIt! evalúa la calidad del agarre.

Aunque sea un simulador principalmente para calcular agarres, no significa que sea lo único que pueda hacer:

- Permite añadir obstáculos para presentar un entorno complejo al robot.
- Incluye una interfaz intuitiva, que permite ser utilizada con Matlab.
- El sistema de detección de colisión y contacto es muy rápido.
- Contiene rutinas de análisis para evaluar la calidad del agarre obtenido.
- Opción de visualización.
- Contiene un generador de trayectorias simple y algoritmos de control para evaluar las fuerzas necesarias en las articulaciones de la mano robótica.

Además de este contenido, GraspIt! también incluye librerías con un brazo robótico, PUMA 560, un robot móvil, XR4000, y manos robóticas entre las que están:

- Paralely jaw gripper
- Barret Hand
- DLR Hand II
- Robonaut Hand
- Rutgas Hand.

Como principales ventajas, este simulador, está bien documentado, incluyendo manuales y tutoriales de uso son, presenta una interfaz de usuario, contiene varios modelos de manos robóticas y funciona tanto en Ubuntu como en Windows.

Y por último, hablaremos de sus principales desventajas como puede ser, la necesidad de utilizar una librería extra para ser utilizado con Matlab y los problemas que surgen de utilizarse con éste, entre los que encontramos, que la interfaz de usuario no funciona ni está en mantenimiento. Otro factor negativo de este simulador es que solo presenta un tipo de calidad de agarre.

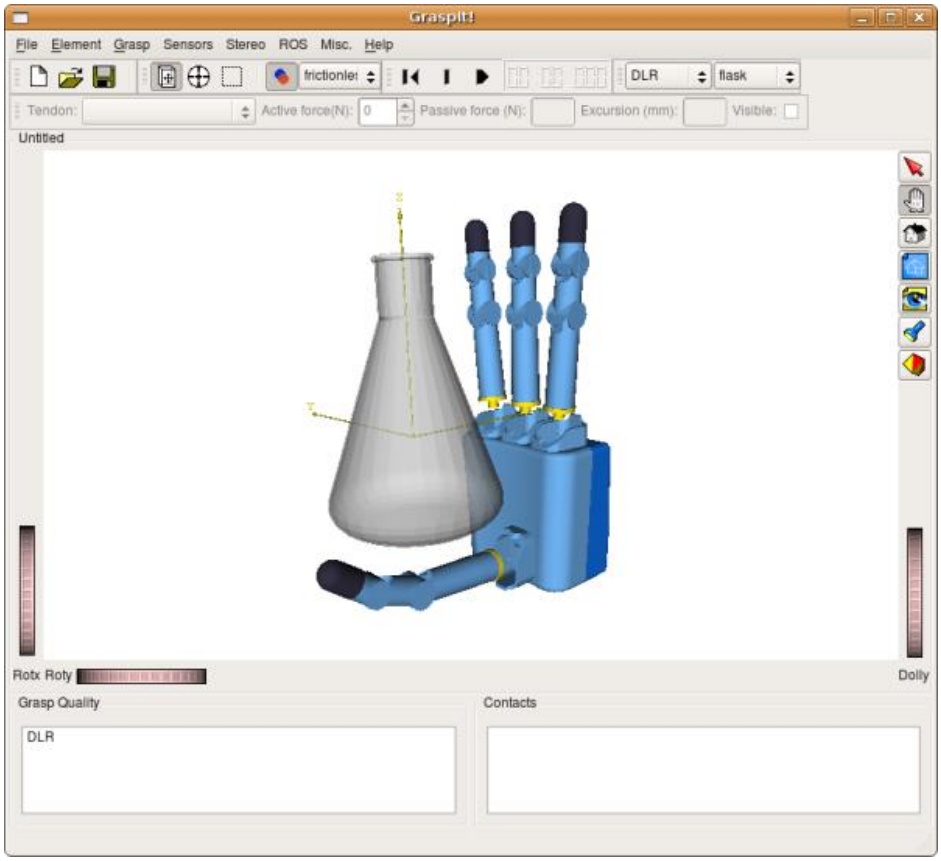

Figura 34. Interfaz de simulador GraspIt!.

OpenRAVE [48]

OpenRAVE es una aplicación de testeo y desarrollo de algoritmos de movimientos para aplicaciones de robots en entornos reales. Su principal uso es el análisis cinemático y el análisis de información geométrica para los planes de movimiento de un robot. Se puede utilizar con una API en Phyton o C++ e incluye varios modelos de robots.

Una de las principales ventajas de esta herramienta es su fácil integración en sistemas robóticos existentes, además, genera automáticamente la documentación para sus distintas versiones, así como incluir tutoriales con los que aprender a manejar el programa y obtener resultados con él.

Como podemos ver, esta aplicación es muy extensa y uno de los módulos que presenta es el módulo de agarre que podemos apreciar en la figura 35.

Este módulo, es un simulador de agarre de objetos y computación de la fuerza del cierre de la mano robótica. OpenRAVE puede simular agarres para cualquier tipo de manos y evalúa la calidad del agarre obtenido.

El cálculo del agarre en este simulador sigue los siguientes pasos:

- Primero muestrea la superficie del objeto para determinar por qué dirección acercarse a él. Al muestrear la superficie, el simulador detecta si hay posibles cavidades en el objeto. También tiene otra forma más sencilla para muestrear la superficie del objeto utilizando la caja delimitadora (*bounding box*).
- Una vez el objeto está muestreado, se guarda el valor de la intersección entre cada haz procedente de cada punto y el objeto.
- Con este valor, se calculan las normales a la superficie, que serán las direcciones que tomará la mano para aproximarse.
- Después, coloca la mano dándole una posición inicial y un valor a sus ángulos.
- La mano se mueve a lo largo de una dirección, generalmente la dirección normal a la palma, usando las calculadas respecto al objeto.
- Una vez el objeto y la mano entran en contacto, los dedos de ésta se van cerrando alrededor del objeto hasta que no se pueden cerrar más.
- Los puntos de contacto entre la mano y el objeto se guardan y se calcula la fuerza ejercida.
- Por último, con el objeto ya agarrado, el simulador añade los puntos de contacto y los puntos de fricción.

Este simulador presenta varias ventajas (que pueden ser la razón de que sea el simulador más utilizado) como puede ser, la extensa documentación que se puede encontrar sobre esta herramienta, entre la que encontramos tutoriales para aprender a utilizar la herramienta. Otra ventaja importante es la extensa librería de robots que presenta debido principalmente, a pertenecer a un programa específico de cálculo de movimientos de robots. Además, el simulador es capaz de muestrear objetos para encontrar cavidades y direcciones óptimas, y todo el proceso de agarre lo realiza en una única función, por lo que con dar unos parámetros se puede obtener el resultado final.

Pero por otro lado, este simulador también tiene sus desventajas, como puede ser, que la documentación no incluya los algoritmos y los conceptos teóricos de éstos para calcular las fuerzas de contacto entre la mano robótica y el objeto, o que al realizar toda la simulación en un único paso, si parte de los parámetros de entrada son erróneos, no se obtendrá ningún resultado.

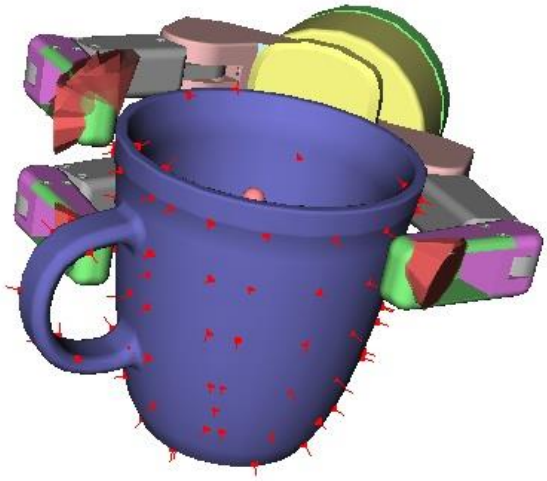

Figura 35. Resultado de Simulación de agarre en OpenRAVE.

SynGrasp [49]

SynGrasp, cuyo simulador de agarre podemos ver en la figura 36, es una *toolbox* de investigación para simular agarres robóticos, está en estado de desarrollo y es para el programa matemático Matlab.

Esta *toolbox* está siendo desarrollada por Monica Malvezzi, Guido Gioioso, Gionata Salvietti y Domenico Prattichizzo en la universidad de Siena desde el año 2012, en el que se publicó su primera versión.

El contenido de la aplicación se puede agrupar en 4 grandes apartados:

- Modelado de manos: SynGrasp ofrece una serie de funciones para implementar una mano robótica, además de incluir varios ejemplos de manos como la DLR Hand II y la Barret Hand, entre otras.
- Modelado de agarre: en este apartado SynGrasp ofrece funciones para elegir, entre otras cosas, la configuración de la mano, añadir puntos de contacto si se conocen, añadir matriz de sinergias, calcular puntos de contacto , calcular la matriz Jacobiana de la mano y la matriz de agarre.
- Análisis de agarre: en esta parte de la *toolbox* están recogidas las funciones que se utilizan para calcular los distintos tipos de calidad de agarre que recoge, además de las funciones necesarias para realizar el análisis cinemático y el análisis de las fuerzas de manipulación.
- Gráficos: todas las funciones utilizadas para representar los objetos, las manos y el agarre entre ellos.

Las principales ventajas que encontramos en SynGrasp son:

- A diferencia de otros simuladores, no necesita conocimientos en programación orientada a objetos, solo conocimientos de programación en Matlab, que es mucho más simple y claro y al pertenecer a Matlab, el uso de matrices es más intuitivo.
- .
- El código puede ser manipulado con facilidad.
- Se pueden crear manos robóticas sin usar las propias funciones incluidas en la *toolbox,* programándolas en Matlab (respetando sus tipos de datos siguiendo el código de las manos de ejemplo incluidas).
- Su librería de funciones es altamente ampliable.
- El código de las funciones de SynGrasp está comentado.

Aunque presenta grandes ventajas, al ser una herramienta en estado de desarrollo también presenta varias desventajas:

- Pese a incluir un manual con todas las funciones y una breve explicación de su uso, se puede afirmar que el contenido de este manual es poco claro y falta información vital para el correcto uso de las funciones explicadas en él.
- SynGrasp, como indica su nombre, utiliza la matriz de Sinergias para algunas funciones, la cual no está incluida en muchos manuales de manos robóticas.
- Hay varias funciones de la herramienta que al estar en estado de desarrollo no funcionan o contienen errores, como puede ser, la interfaz de usuario que viene incluida en la herramienta.
- Para crear objetos propios utilizando las funciones de la propia *toolbox,* utiliza datos proporcionado por otras funciones incluidas SynGrasp, por lo que no se pueden crear fácilmente.
- SynGrasp solo incluye objetos sencillos, esfera, prisma y cilindro por lo que muchas funciones solo tiene en cuenta estos objetos y no funcionan para otros objetos creados a partir de la propia herramienta.
- La precisión y calidad de los agarres es baja comparado con la de otros simuladores.

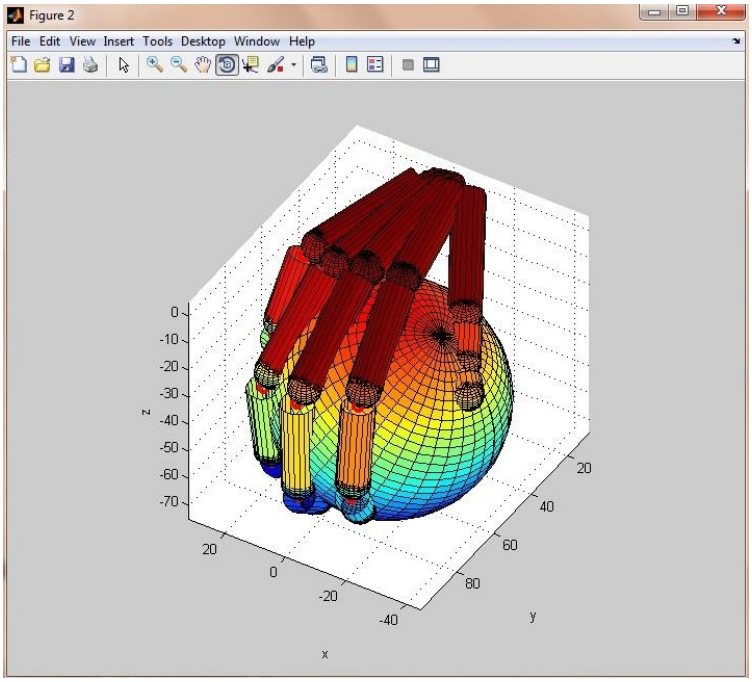

Figura 36. Simulador de agarres SynGrasp.

Pese a ello la *toolbox* SynGrasp es el simulador de agarre elegido para este proyecto, ya que, la facilidad de añadir código que presenta y la facilidad en el manejo de matrices es mucho mayor que en el resto de simuladores. Otra de las razones para elegir esta *toolbox* es que este proyecto tiene, entre otros objetivos, mejorar la librería y funcionalidad de la misma.

La descripción y el funcionamiento del método seguido para el agarre de objetos y la descripción y tipos de calidad de agarre serán analizados en apartados posteriores.

5 Simulación y generación de agarres.

Con el simulador ya elegido, procederemos a realizar las mejoras. Estas mejoras, se centran en implementar la mano elegida, Gifu Hand III, permitir el uso de cualquier tipo de objeto, si es introducido en formato STL y crear un nuevo método para calcular los puntos de agarre de estos objetos.

5.1 Creación de la mano Gifu III con la herramienta SynGrasp.

Comprobando los ejemplos de mano que incluye SynGrasp, se pueden encontrar los parámetros y la estructura de los parámetros que conforman las manos.

El primer paso para crear la mano Gifu Hand III, es conocer en su totalidad sus dimensiones, que podemos encontrar en las figuras 29 y 30. Una vez conocidas todas las dimensiones de la mano, es necesario calcular los parámetros d-h de esta.

5.1.1 Cálculo de los parámetros d-h y de la matriz de transformación.

Denavit-Hatenberg es un método matricial que permite conocer el sistema de coordenadas ligado a un eslabón i (que forma parte de una cadena articulada) a partir de cuatro parámetros, con el fin de determinar las ecuaciones cinemáticas de la cadena completa [50].

Los cuatro parámetros que definen las relaciones existentes entre los ejes consecutivos i-1 e i son:

- **Ángulo θ**, este parámetro recoge el ángulo de rotación girado alrededor del eje z_{i-1} para situar el eje x_{i-1} en la misma posición que el eje x_i.
- **Distancia d**, es la distancia a lo largo del eje z_{i-1}, desde el origen del sistema de coordenadas del eslabón i-1 hasta la normal común con el eslabón i.
- **Distancia a**, es la distancia a lo largo de la normal común, o radio de rotación respecto al eje z_{i-1}.
- **Ángulo α**, es el ángulo alrededor del eje x_i para colocar el eje z_i en la posición deseada respecto al eje z_{i-1}.

Con estos parámetros, se obtienen las matrices de transformación homogéneas parciales, $^{i-1}\mathbf{A_i}$ que se crean de la siguiente manera:

$$^{i-1}\mathbf{A_i}= \begin{bmatrix} Cos\theta i & -Cos\alpha i\ Sin\theta i & Sin\alpha i\ Sin\theta i & ai\ Cos\theta i \\ Sin\theta i & Cos\alpha i\ Cos\theta i & -Sin\alpha i\ Cos\theta i & -ai\ Sin\theta i \\ 0 & Sin\alpha i & Cos\alpha i & di \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para calcular los 4 parámetros d-h y la matriz de transformación existe un algoritmo que consta de un total de 16 pasos [9]:

- **DH1.** Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.
- **DH2.** Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n.
- **DH3.** Localizar el eje de cada articulación. Si esta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.
- **DH4.** Para i de 0 a n-1, situar el eje z_i, sobre el eje de la articulación i+1.
- **DH5.** Situar el origen del sistema de la base S_0 en cualquier punto del eje z_0. Los ejes x_0 e y_0 se situaran de modo que formen un sistema dextrógiro con z_0.
- **DH6.** Para i de 1 a n-1, situar el sistema S_i (solidario al eslabón i) en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i. Si ambos ejes se cortasen se situaría S_i en el punto de corte. Si fuesen paralelos S_i se situaría en la articulación i+1.
- **DH7.** Situar x_i en la línea normal común a z_{i-1} y z_i.
- **DH8.** Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i.
- **DH9.** Situar el sistema S_n en el extremo del robot de modo que Zn coincida con la dirección de Zn-1 y Xn sea normal a Zn-1 y Zn.
- **DH10.** Obtener θ_i como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos.
- **DH11.** Obtener D_i como la distancia, medida a lo largo de z_{i-1}, que habría que desplazar S_{i-1} para que x_i y x_{i-1} quedasen alineados.
- **DH12.** Obtener a_i como la distancia medida a lo largo de x_i (que ahora coincidiría con x_{i-1}) que habría que desplazar el nuevo S_{i-1} para que su origen coincidiese con S_i.
- **DH13.** Obtener α_i como el ángulo que habría que girar en torno a x_i (que ahora coincidiría con X_{i-1}), para que el nuevo (S_{i-1}) coincidiese totalmente con S_i.
- **DH14.** Obtener las matrices de transformación ${}^{i-1}A_i$.
- **DH15.** Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot $T = {}^{0}A_1, {}^{1}A_2 \ldots {}^{n-1}A_n$.
- **DH16.** La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las n coordenadas articulares.

5.1.2 Valores obtenidos y límites.

El cálculo de estos parámetros en una mano robótica solo se necesita en los dedos, (partiendo del punto de unión de palma y dedo como origen) por lo que, una mano robótica tendrá un total de 5 tablas de parámetros d-h y 5 matrices de transformación.

La mano Gifu Hand III utiliza una nomenclatura especial de parámetros d-h (la notación de Denavit-Hatemberg modificada introducida en 1986 por Craig [50]), debido a esto, los parámetros d-h facilitados en el manual de la mano Gifu Hand III no son válidos, pues en SynGrasp se utiliza la nomenclatura tradicional.

Como la estructura cinemática de la mano Gifu Hand III, es la misma que la de la mano de ejemplo paradigmatic de 20 GDL (incluida en la librería de SynGrasp), se han utilizado los parámetros d-h incluidos en ésta para crear nuestra mano, modificando únicamente los valores *a* y *d*.

En las tablas 2 y 3, se muestra el valor de los parámetros d-h, necesarios para la creación de la mano, obtenidos para cada dedo y con el movimiento correspondiente a cada eje:

Eje i	**α**	**a**	**θ**	**d**	**Movimiento**
1	-π/2	0	0	0	Abducción/aducción
2	0	70,5	0	0	Flexión/extensión
3	0	50	0	0	Flexión/extensión
4	0	36,2	0	0	Flexión/extensión

Tabla 2. Parámetros D-H para el pulgar.

Eje i	**α**	**a**	**θ**	**d**	**Movimiento**
1	-π/2	0	0	0	Abducción/aducción
2	0	70,5	0	0	Flexión/extensión
3	0	32,5	0	0	Flexión/extensión
4	0	32	0	0	Flexión/extensión

Tabla 3. Parámetros D-H para el resto de los dedos.

Una vez obtenidos los parámetros d-h, se puede calcular las matrices de transformación asociadas a cada dedo, que en el simulador, forman parte de una matriz tridimensional llamada base.

Estas matrices, vienen incluidas en la hoja de datos de la Gifu Hand III, pero los datos recogidos en ellas no son válidos para crear una mano funcional en el simulador SynGrasp. Esto se debe, principalmente, a una variación en los ejes de coordenadas de la mano respecto a los que utiliza SynGrasp, y al problema de utilizar distintos parámetros d-h.

Si se usan estas matrices de transformación, la mano creada aparece orientada respecto a otros ejes y los movimientos realizados por los dedos no son los correspondientes (abducción/aducción, extensión/flexión).

Debido a esto, se decidió calcular estas matrices (los parámetros d-h ya eran conocidos, los ángulos de orientación de los dedos se pueden obtener del manual, y las coordenadas de los primeros puntos de cada dedo se consiguen de las matrices de transformación incluidas en la mano) obteniéndose los siguientes datos:

- Matriz de transformación para el pulgar:

$$\text{Base } \{1\} = \begin{bmatrix} -0{,}6480 & -0{,}1026 & -0{,}7547 & -38{,}6 \\ 0{,}2587 & -0{,}9616 & -0{,}0913 & 13{,}9 \\ -0{,}7164 & -0{,}2544 & 0{,}6497 & -21{,}5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matriz de transformación para el dedo índice:

$$\text{Base } \{2\} = \begin{bmatrix} -0{,}1736 & -0{,}9848 & 0 & -38 \\ 0{,}9848 & -0{,}1736 & 0 & 100{,}4 \\ 0 & 0 & 1 & -7{,}5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matriz de transformación para el dedo medio:

$$\text{Base } \{3\} = \begin{bmatrix} 0 & -1 & 0 & -12 \\ 1 & 0 & 0 & 116{,}4 \\ 0 & 0 & 1 & -7{,}5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matriz de transformación para el dedo anular:

$$\text{Base } \{4\} = \begin{bmatrix} 0{,}1736 & -0{,}9848 & 0 & 14 \\ 0{,}9848 & 0{,}1736 & 0 & 108{,}4 \\ 0 & 0 & 1 & -7{,}5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matriz de transformación para el dedo meñique:

$$\text{Base } \{5\} = \begin{bmatrix} 0{,}342 & 0{,}9397 & 0 & 40 \\ 0{,}9397 & 0{,}342 & 0 & 88{,}4 \\ 0 & 0 & 1 & -7{,}5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Por último, se añaden los límites máximos y mínimos de movimiento para cada eje de la mano, cuyos valores, se encuentran en la hoja de datos, estos se pueden ver en la tabla 4.

	Operating range [deg]	
	Thumb	Finger
1st joint	-28∼28	-20∼20
2nd joint	-10∼90	-10∼90
3rd joint	-10∼90	-10∼90
4th joint	-10∼90	-10∼90

Tabla 4. Rangos de operación para las articulaciones de cada dedo.

Con los parámetros d-h, las matrices de transformación calculadas y con los rangos para cada articulación establecidos, la mano finalmente es creada utilizando el resto de funciones incluidas en la mano de ejemplo paradigmatic, dando como resultado la mano que observamos en las figuras 37, 38 y 39. Es por este motivo, que la función encargada de crear la mano recibe el nombre de SGgifuIII_paradimgatic.

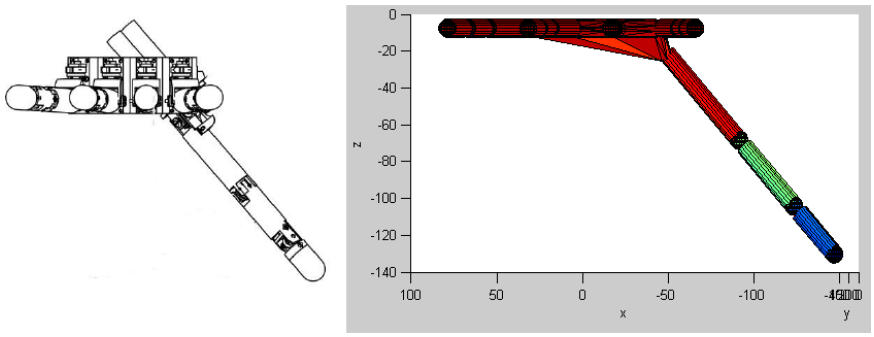

Figura 37. a) Planta de la mano Gifu Hand III. b) Planta de la mano Gifu hand III obtenida.

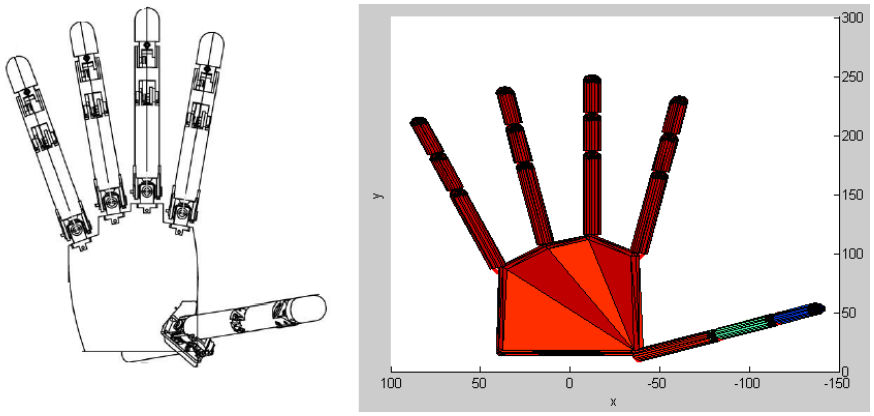

Figura 38. a) Alzado de la mano Gifu Hand III. b) Alzado de la mano Gifu hand III obtenida.

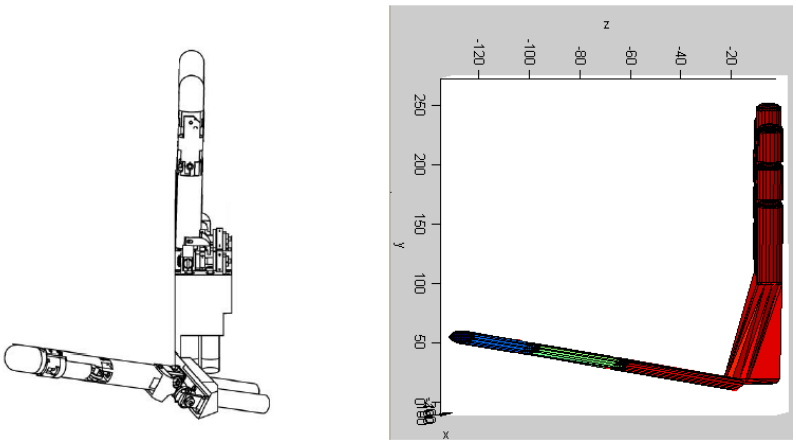

Figura 39. a) Perfil de la mano Gifu Hand III. b) Perfil de la mano Gifu hand III obtenida.

La estructura de Matlab generada, consta de los siguientes campos:

- Hand.F: este campo, almacena las estructuras que contienen la información correspondiente a cada dedo de la mano y también, se divide en varios campos más:
    - F.n: indica el número de articulaciones que contiene el dedo seleccionado.
    - F.DHpars: matriz con los parámetros d-h correspondientes al dedo seleccionado.
    - F.base: matriz de transformación correspondiente al dedo seleccionado.
    - F.q: vector columna, con tantas filas como articulaciones tenga el dedo seleccionado (en el caso de nuestra mano 4), en cada fila se almacena el valor, en radianes, que es el correspondiente al giro de la articulación (inicialmente todas valen 0).
    - F.qin: vector columna, con tantas filas como articulaciones tenga el dedo seleccionado, cada fila, contiene el número con el orden correspondiente de la articulación (de 1 a 4 en el caso de nuestra mano).
    - F.joints: matriz con las coordenadas de cada articulación y con las coordenadas del extremo del dedo seleccionado.
- Hand.n: indica el número de dedos que tiene la mano creada (5).
- Hand.m: indica el número de articulaciones que tiene la mano (20).
- Hand.q: vector columna, con tantas filas como articulaciones tenga la mano, (20), y almacena el valor, en radianes, que es el correspondiente al giro de la articulación (inicialmente 0).
- Hand.qin: vector columna, con tantas filas como articulaciones tenga la mano, cada fila, contiene el valor al dedo asociado al que pertenece la articulación correspondiente (de 1 a 5 para nuestra mano).
- Hand.qinf: vector columna, con tantas filas como articulaciones tenga la mano, cada fila, contiene el número con el orden correspondiente de la articulación en cada dedo (de 1 a 4 en el caso de nuestra mano).

- Hand.ctype: indica la dureza de los dedos de la mano, 1 para dureza alta, 2 para dureza media y 3 para dureza baja, este valor en nuestra mano es 1.
- Hand.ftips: matriz que contiene las coordenadas de la punta de cada dedo de la mano.
- Hand.S: matriz de sinergias asociada a la mano.
- Hand.cp: matriz que almacena los puntos de contacto con un objeto.
- Hand.Kp: matriz utilizada para calcular los puntos de contacto.
- Hand.Kz: matriz utilizada para calcular los puntos de contacto.
- Hand.H: matriz homogénea de la mano.
- Hand.J: matriz jacobiana de la mano.
- Hand.JS: matriz jacobiana subactuada de la mano.
- Hand.limit: matriz que contiene los límites de cada articulación, cada fila contiene los limites inferiores y máximos en ese orden.
- Hand.active: vector columna, con tantas filas como articulaciones tenga la mano, en cada fila, se almacena un 1 si la articulación correspondiente se utiliza o un 0 si no se permite su movimiento.
- Hand.type: nombre de la mano.
- Hand.Jtilde: matriz jacobiana calculada para la mano.

Como se puede observar, hasta este punto la mano Gifu Hand III que se ha creado, no es exactamente como debe ser, pues el número de GDL que presenta no es 16, sino 20. Esto es debido a que no se pueden incluir los 4 GDL que sobran en el campo *Hand.active*, ya que éste, bloquea el movimiento de estas articulaciones y ese no es nuestro caso (en la mano Gifu Hand III, las últimas articulaciones de todos los dedos, salvo el pulgar, se mueven junto con las articulaciones anteriores a estas).

Para solucionar este problema, se ha modificado la función que mueve la mano, SGmoveHand, para que estas últimas articulaciones se liguen a las anteriores, moviendo cada una el mismo valor que sus articulaciones anteriores.

5.1.3 Funciones añadidas.

Una vez creada la mano, se ha ampliado la librería de funciones de SynGrasp. El fin de estas funciones, es comprobar el correcto funcionamiento de la mano creada.

Se han creado un total de 3 funciones para este fin, el funcionamiento de estas se puede apreciar en la figura 40:

- SGcheckmove: esta función, pide como parámetro de entrada una mano y devuelve una secuencia de video en el gráfico de Matlab, mostrando, en orden, el movimiento de todas las articulaciones, sin tener en cuenta ninguna restricción. La utilidad de esta función es permitir comprobar el correcto movimiento (flexión/extensión, abducción/aducción) que debe realizar cada articulación.

- SGcheckmovejoint: esta función, tiene el mismo objetivo que la anterior, pero añade la opción de elegir qué articulación probar, para evitar ver la secuencia completa de la mano. La utilidad de esta función, se centra en comprobar una determinada articulación que tenga algún tipo de problemas por su dificultad (útil para el dedo pulgar).
- SGcheckmove_withlimits: esta función, es como la función SGcheckmove, solo que incluye los límites de cada articulación para poder comprobar si son los correctos. Los límites no se han incluido en la función SGcheckmove para analizar cada problema por separado, por lo que es recomendable utilizar esta función una vez comprobado el correcto funcionamiento de la mano.

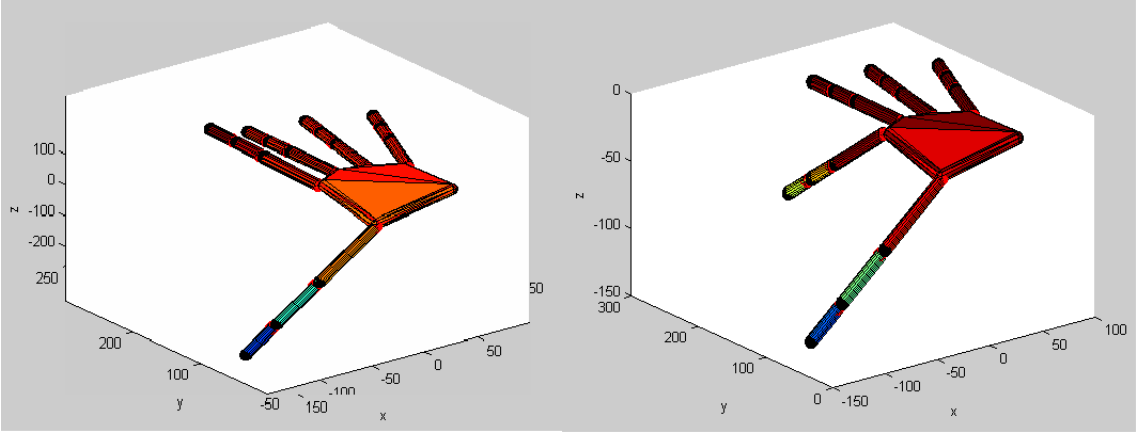

Figura 40. a) Inicio de comprobación del movimiento del dedo anular de la mano Gifu Hand III. b) Comprobación del movimiento de la segundo articulación (flexión/extensión) del dedo anular de la Gifu Hand III.

5.2 Representación de un objeto con formato STL.

STL (STereoLithography), es un formato de archivo procedente del software de diseño CAD, creado por 3DSystems y utilizado para la creación de objetos 3D. Este tipo de formato, solo almacena los datos referentes a geometría, no almacena datos referentes a textura o color [51]. En el formato STL, se pueden almacenar los datos de dos formas posibles: en ASCII o en binario. Generalmente, se suele utilizar el método binario, pues el archivo es más compacto de esta manera.

Un archivo STL, describe una superficie formada por triángulos, de los que se almacenan sus 3 vértices y el vector normal (usando el método de la mano derecha), en un sistema cartesiano, y sin almacenar información correspondiente a la escala o unidades de medida.

En ambas versiones, solo se utilizan 12 puntos por triángulo (3 para cada vértice y 3 para la normal del triángulo) y la normal debe ser unitaria y apuntando hacia el exterior del objeto.

En principio, este formato no almacena información sobre el color, pero existen variaciones para la versión binaria del formato STL para incluir información sobre el color.

5.2.1 Funciones y pasos seguidos.

La función SGstl, es la encargada de crear un objeto válido para utilizarse en la simulación de agarre de SynGrasp. Esta función, realiza varios pasos para convertir los datos proporcionados por el objeto, en estructuras válidas:

El primer paso, es poder trabajar en Matlab con objetos en formato STL, para ello, se ha utilizado una función ya existente (READ_stl), encargada de obtener y almacenar los puntos y normales que forman el objeto, independientemente de si este ha sido definido en ASCII o en binario. El objeto obtenido, como el de la figura 41, aun no presenta una estructura válida para ser utilizado por SynGrasp.

Los parámetros que devuelve esta función son:

- Una matriz de tamaño Nx3x3: en esta matriz, se almacenan las coordenadas de los vértices que forman cada triángulo. N, es el número de triángulos y cada triángulo tiene una matriz 3x3 donde las columnas son las coordenadas x, y, z de las filas, que son los vértices del triángulo.
- Una matriz Nx3: en ésta, se almacenan las normales de cada triángulo que forma el objeto. Esta matriz, tiene tantas filas como triángulos tenga el objeto, y las columnas corresponden con las coordenadas x, y, z.

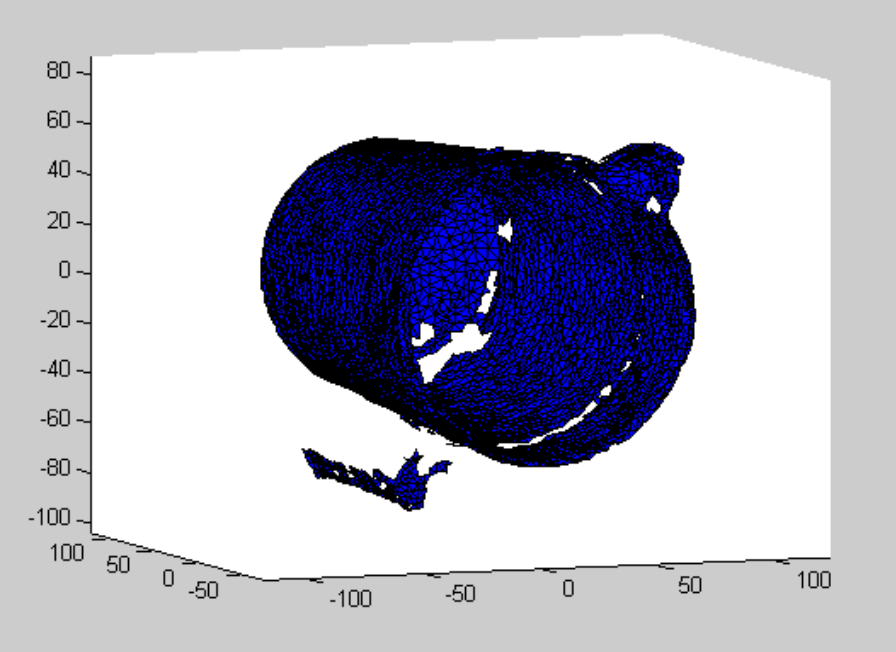

Figura 41. Representación en Matlab de un objeto obtenido por la función READ_stl.

Los valores obtenidos, son introducidos en otra función, CONVERT_meshformat, con la que reducir el número de puntos totales (eliminando los vértices repetidos que aparecen entre triángulos contiguos), y obteniendo las caras del objeto.

Una vez hemos conseguido almacenar los datos, se ha decidido trabajar con la *bounding box* del objeto (la *bounding box* o cuadro delimitador, es el prisma invisible de menor tamaño que contiene al objeto), para facilitar futuros cálculos.

Con el fin de calcular la *bounding box*, se ha creado otra función, SGboundingbox, que utilizando las coordenadas de los puntos que forman el objeto, crea la estructura de la misma, como vemos en la figura 42.

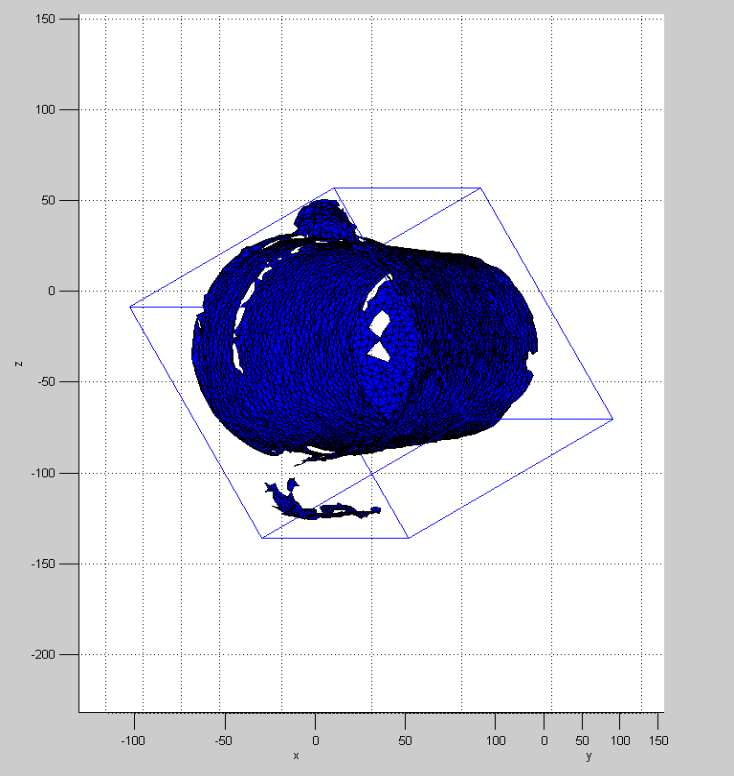

Figura 42. Objeto con bounding Box.

Para lograrlo, se han analizado todos los puntos y almacenado los que contienen las coordenadas mínima y máxima de cada eje, conociendo, de esta manera, las tres dimensiones del prisma.

Una de las utilidades de calcular la *bounding box*, es la de poder calcular el centro de los objetos, aproximándolo al centro de ésta. La función encargada de esto es SGcenterOfBoundingbox.

La principal razón para calcular el centro, es la necesidad de éste para poder mover y girar la pieza, ya que hasta el momento, solo se podía representar el objeto en una posición inicial y coordenadas fijas (el origen de los ejes de coordenadas), es decir, gracias al centro, se tiene un punto de referencia, que se puede mover a la posición deseada y desde el cual pueden rotar y trasladarse el resto de puntos que forman el objeto.

Finalmente, se añaden el resto de parámetros necesarios para que el objeto tenga una estructura válida para ser utilizada por el resto de la librería de SynGrasp.

Los parámetros que tiene el objeto, inicialmente, son:

- type: parámetro utilizado por SynGrasp para distinguir el tipo de objeto con el que está trabajando y así utilizar los algoritmos correspondientes a éste.
- name: nombre del fichero STL.
- Htr: matriz de transformación, introducida como parámetro a la función, con esta se pude elegir donde situar el centro del objeto y cuanto girarlo.

- coordNORMALS: matriz con las coordenadas normales asociadas a los triángulos que forman el objeto.
- CoordVERTICES: matriz con las coordenadas de los vértices que forma cada triangulo.
- center: vector que contiene las coordenadas del centro del objeto.
- bounding: matriz que contiene las coordenadas mínimas y máximas de cada eje de coordenadas, obtenidas de los puntos del objeto.
- centrobounding: vector con las coordenadas del centro del objeto, calculado a partir de la *bounding box*. Con la diferencia entre este vector y el vector center se calcula el valor necesario para trasladar todos los puntos, de la posición original a la establecida por el parámetro Htr.
- faces: matriz con las caras del objeto.
- vertices: matriz con las coordenadas de los puntos (sin repetir) que forman el objeto.
- boundingbox: estructura que forma la *bounding box*, está compuesta por:
    - Htr: matriz de transformación, que permite trasladar y girar la *bounding box* la misma cantidad que el objeto.
    - Center: coordenadas del centro de la *bounding box.*
    - dim: matriz con las dimensiones del prisma.
    - faces: estructura con las coordenadas de las caras que forman el prisma.
- maxdim: mitad de la máxima dimensión de la *bounding box*, se utiliza en las funciones de cálculo de agarre, para colocar la mano a una distancia aceptable respecto al objeto.

Con todos estos parámetros, esta estructura ya es válida para ser utilizada por SynGrasp, y por tanto, el objeto puede ser modificado por las funciones de la *toolbox*, añadiendo más parámetros que utiliza el propio simulador:

- cp: matriz con los puntos de contacto con la mano.
- H: matriz homogénea.
- G: matriz de agarre resultado de multiplicar Gtilde por la matriz homogénea.
- Gtilde: matriz de agarre calculada a partir de los puntos de contacto.
- Kc: parámetros Kc de la mano.
- normals: direcciones normales de la superficie de contacto.
- base: matriz de transformación recalculada.

Ya que Syngrasp ha podido añadir estos parámetros, se comprueba que la estructura creada para objetos con formato STL funciona como una más de la librería.

5.3 Agarre de objetos.

Una vez implementada la mano Gifu Hand III y creado un nuevo tipo de objeto con el que trabajar en la *toolbox*, es necesario comprobar si estas implementaciones funcionan en la simulación de agarre de SynGrasp.

Los pasos para realizar esta comprobación son, probar si nuestra mano coge los objetos ya incluidos en la herramienta (esfera, cilindro y cubo) y en caso afirmativo, ver si el resultado es parecido, en cuanto a calidad, al agarre obtenido por las manos ya incluidas en SynGrasp. También comprobar si se realiza el agarre del objeto stl con nuestra mano y con las incluidas en la herramienta.

Para realizar estas pruebas, se han utilizado dos tipos de manos proporcionadas por SynGrasp, una mano no antropomórfica de 3 dedos y la mano antropomórfica paradigmatic. Esta última, es más importante en nuestra comparación de agarres, por su parecido con la Gifu Hand III.

La función utilizada para este cometido es, GraspPlannerExample, que ha sido modificada para incluir al nuevo tipo de objetos y a la mano Gifu Hand III, así como nuevas funciones para éstos. Esta función, pide al usuario el tipo de mano con el que se quiere trabajar (Paradigmatic, 3Fingered, Modular o Gifu Hand III) y la crea. Acto seguido, define la matriz Htr (matriz de transformación) del objeto, para indicar donde se colocará éste y en qué posición.

Una vez creada la mano y la matriz Htr, el programa pide el número de agarres que se van a efectuar (el simulador de agarres en SynGrasp se basa en realizar varias iteraciones, con el fin de encontrar el mejor agarre posible) y el objeto que se va a utilizar (esfera, cilindro, cubo o stl). Si el objeto es stl, además pedirá el nombre del archivo que utilizará para crearlo. El programa, crea el objeto y lo coloca en la posición indicada por la matriz Htr.

Finalmente, el programa pide el tipo de calidad de agarre que se va a usar en la ejecución, de un total de 5 tipos distintos que incluye la herramienta (mev, gii, msv, dtsc, y uot, que serán explicados más adelante).

Con todos los datos recopilados, el programa llama a la rutina principal encargada de la simulación, SGgraspPlanner.

Lo primero que realiza esta función, es indicar qué articulaciones de la mano elegida se permite mover, y el valor de exactitud que tendrá el agarre. Después, se genera una nube de puntos alrededor del objeto con posición y orientación aleatoria, con tantos puntos como iteraciones se hayan pedido (estos puntos indican donde se situará el centro de la mano en cada iteración).

Para colocar la mano en estos puntos, se calcula (mediante una función) el centro de la mano y se le añade un *offset* a ésta, para evitar que entre en contacto con el objeto antes de la simulación, dando el resultado de la figura 43.

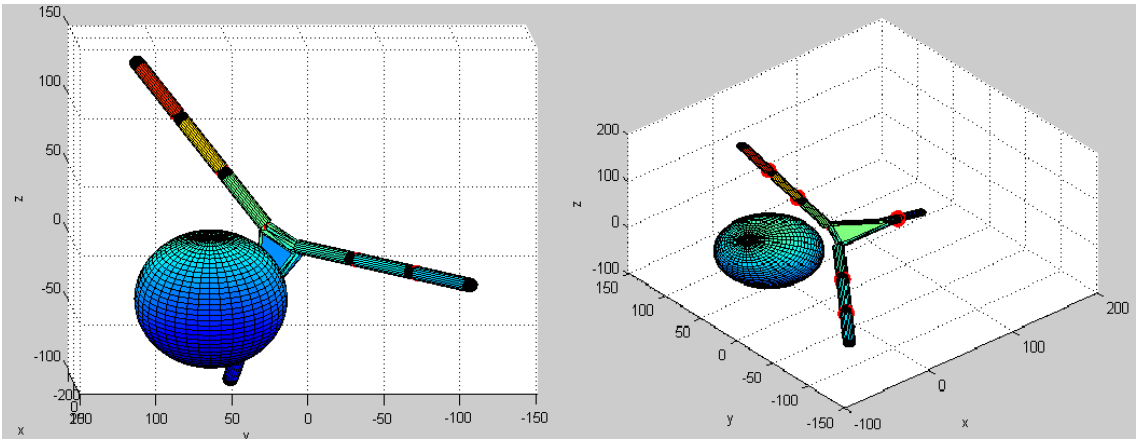

Figura 43. Pre-agarre de una esfera con la mano 3fingered.

Para cada iteración, una vez que la mano esté colocada en el punto indicado, se procede a cerrar la mano con la función, SGcloseHand. Esta función, muestra el algoritmo que utiliza SynGrasp para agarrar objetos.

SGcloseHand, se encarga de cerrar la mano, moviéndola en pequeños incrementos hasta un número máximo de iteraciones o hasta que se haya llegado a la posición final.

La mano, se va cerrando articulación a articulación (de las que se ha permitido movimiento), empezando con las articulaciones del pulgar (en orden) y acabando con las del meñique, hasta alcanzar el objeto o llegar al límite de rango que ofrece la articulación (si se llega a éste antes de alcanzar el número máximo de incrementos).

En cada incremento, la rutina comprueba si se ha establecido contacto con el objeto utilizando la función, SGcontactDetection. Para ello, ésta almacena las coordenadas correspondientes a la articulación evaluada y las coordenadas de la siguiente articulación (en caso de la última articulación de cada dedo, utiliza como último punto la punta de éstos) de tal forma, que queda definido el rango del eslabón a analizar.

Con el eslabón definido, se analiza si existen puntos de contacto con el objeto, de forma distinta según el tipo de objeto que se esté utilizando (los cálculos para cada objeto, serán analizados posteriormente). Dependiendo de si se ha producido contacto entre el objeto y la mano, se devuelve un valor que indicará al programa principal si seguir realizando incrementos o cambiar de articulación y generar los puntos de contacto.

Si el valor indica que hay puntos de contacto, se crean las matrices de puntos de contacto, tanto en la mano como en el objeto.

Estas matrices, tienen la estructura que podemos observar en la figura 44, donde cada columna es un punto de contacto. Las primeras tres filas, indican las coordenadas del punto de contacto, la cuarta fila, indica el dedo que lo está produciendo, la quinta fila, indica el eslabón del dedo que está en contacto, la sexta es un valor entre 0 y 1 que muestra que parte del eslabón está en contacto (0 si es el inicio del eslabón, 1 si es el extremo) y la séptima fila, indica el tipo de contacto que se está produciendo (1 dureza alta, 2 dureza media, 3 dureza baja).

hand_c × hand_c.cp ×

hand_c.cp <7x3 double>

	1	2	3	4
1	0	-29.5875	-13.3560	
2	31.5694	49.2610	105.2308	
3	-57.3539	2.7286	-56.3598	
4	1	2	3	
5	2	3	3	
6	1	1	1	
7	1	1	1	
8				

Figura 44. Matriz con puntos de contacto entre la mano 3fingered y una esfera.

Este proceso, se realiza hasta completar el número de agarres deseado y finalmente, muestra y almacena todos los valores (mano y objeto) del mejor agarre obtenido, según la calidad de agarre elegida, con los campos de puntos de contacto rellenos.

5.3.1 Cálculo de los puntos de contacto con una esfera.

Para calcular los puntos de contacto entre una mano y una esfera, SynGrasp posee su propia función, SGintSegSph, que utiliza la siguiente ecuación para realizar dicho cálculo:

$$A\alpha^2 + B\alpha + C = 0 \quad (1)$$

donde:

- α, indica el valor entre 0 y 1 que se devolverá en caso de que exista contacto, o NaN en caso de no existir dicho contacto.

- A, es la distancia euclídiana al cuadrado entre los puntos 1 y 0 del eslabón:

$$A = (x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2 \quad (2)$$

  - X_1, Y_1, Z_1: son las coordenadas del final del eslabón.
  - X_0, Y_0, Z_0: son las coordenadas del principio del eslabón del eslabón.

- B, es el resultado de:

$$B = 2 \cdot ((x_1 - x_0) \cdot (x_0 - x_c) + (y_1 - y_0) \cdot (y_0 - y_c) + (z_1 - z_0) \cdot (z_0 - z_c)) \quad (3)$$

  - X_1, Y_1, Z_1: son las coordenadas del final del eslabón.
  - X_0, Y_0, Z_0: son las coordenadas del principio del eslabón.
  - X_c, Y_c, Z_c: son las coordenadas del centro de la esfera.

- C, obtenido como:

$$C = (x_0 - x_c)^2 + (y_0 - y_c)^2 + (z_0 - z_c)^2 - R^2 \quad (4)$$

  - X_1, Y_1, Z_1: son las coordenadas del final del eslabón.
  - X_0, Y_0, Z_0: son las coordenadas del principio del eslabón del eslabón.
  - X_c, Y_c, Z_c: son las coordenadas del centro de la esfera.
  - R: es el radio de la esfera.

En las figuras 45 y 46, se aprecia el correcto funcionamiento de la simulación de agarres de esferas con las manos ya proporcionadas por la herramienta. En la figura 47 se comprueba que la mano Gifu Hand III también funciona para este tipo de agarres y no presenta ningún tipo de problemas.

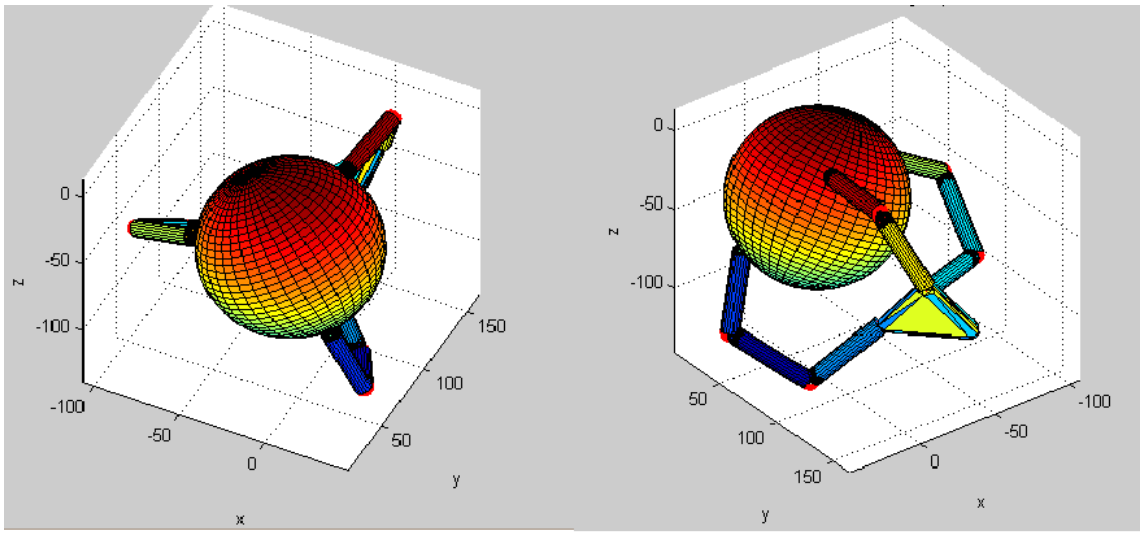

Figura 45. Simulación de agarre de esfera con la mano no antropomórfica 3fingered.

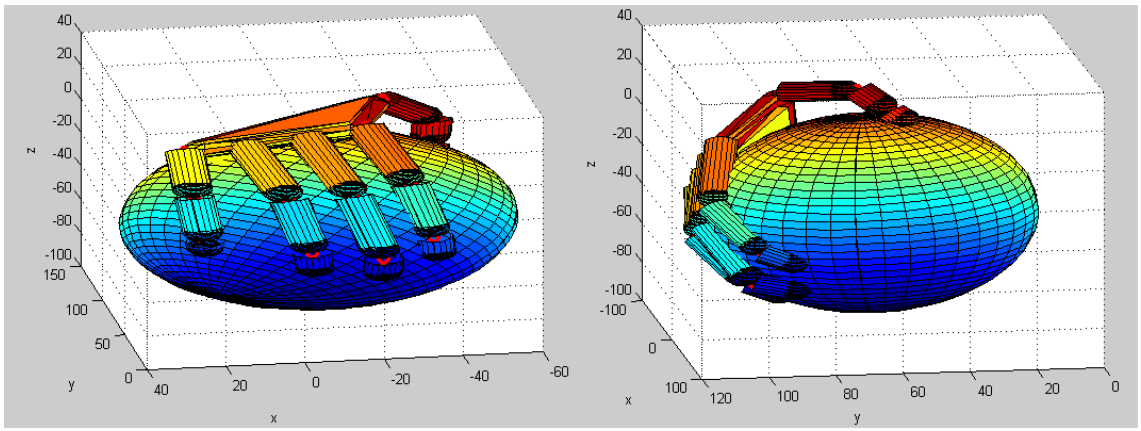

Figura 46. Simulación de agarre de esfera con la mano antropomórfica paradigmatic.

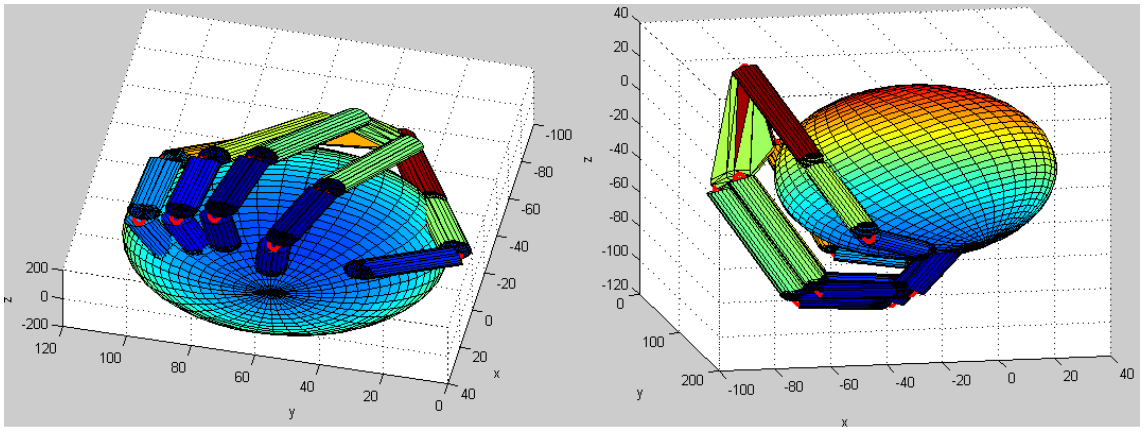

Figura 47. Simulación de agarre de esfera con la mano antropomórfica Gifu Hand III.

5.3.2 Cálculo de los puntos de contacto con un cilindro.

Para calcular los puntos de contacto entre una mano y un cilindro, SynGrasp posee su propia función, SGintSegCyl, que utiliza la siguiente ecuación para realizar dicho cálculo:

$$A\alpha^2 + B\alpha + C = 0 \quad (5)$$

donde:

- α, indica el valor entre 0 y 1 que se devolverá en caso de que exista contacto, o NaN en caso de no existir dicho contacto.

- A, se obtiene de la siguiente ecuación

$$A = (x_1 - x_0)^2 + (y_1 - y_0)^2 \qquad (6)$$

  - X_1, Y_1: son las coordenadas del final del eslabón multiplicadas por la inversa de la matriz de transformación del cilindro.
  - X_0, Y_0: son las coordenadas del principio del eslabón del eslabón.

- B, es el resultado de:

$$B = 2 \cdot ((x_1 - x_0) \cdot (x_0 - x_c) + (y_1 - y_0) \cdot (y_0 - y_c)) \qquad (7)$$

  - X_1, Y_1: son las coordenadas del final del eslabón multiplicadas por la inversa de la matriz de transformación del cilindro.
  - X_0, Y_0: son las coordenadas del principio del eslabón del eslabón.
  - X_c, Y_c, valen 0.

- C, obtenido como:

$$C = (x_0 - x_c)^2 + (y_0 - y_c)^2 - R^2 \quad (8)$$

  - X_1, Y_1: son las coordenadas del final del eslabón.
  - X_0, Y_0: son las coordenadas del principio del eslabón del eslabón.
  - X_c, Y_c: son las coordenadas del centro de la esfera.
  - R: es el radio del cilindro.

En las figuras 48 y 49, se aprecia el funcionamiento de la simulación de agarres de cilindros con las manos ya proporcionadas por la herramienta. En la figura 49 ya se empieza a observar la carencia del programa en la simulación de agarre de objetos más complejos. En este caso, la mano proporcionada por la propia herramienta, atraviesa varias partes del objeto. Esto se debe, a que este programa se centra más en el aumento del número de iteraciones que en mejorar el algoritmo de contacto, para lograr una mejor simulación. Por todo esto, que nuestra mano atraviese objetos como la anterior es

problema del simulador y no de la programación realizada. En la figura 50 se comprueba que la mano Gifu Hand III también funciona como las anteriores para este tipo de agarres.

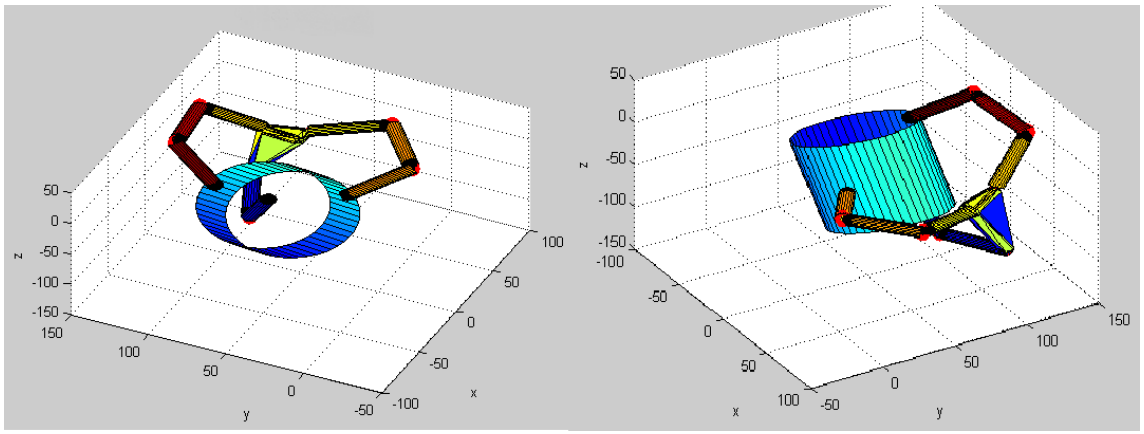

Figura 48. Simulación de agarre de un cilindro con la mano no antropomórfica 3fingered.

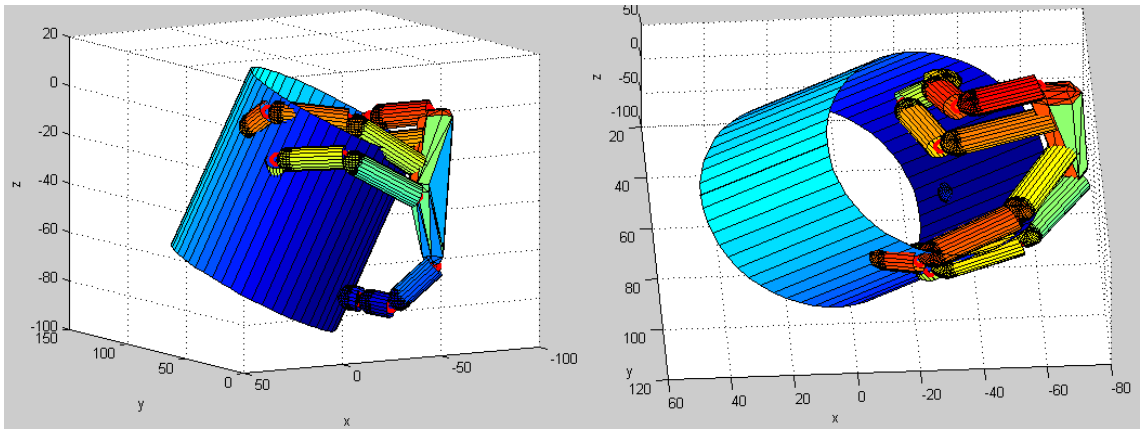

Figura 49. Simulación de agarre de un cilindro con la mano antropomórfica paradigmatic.

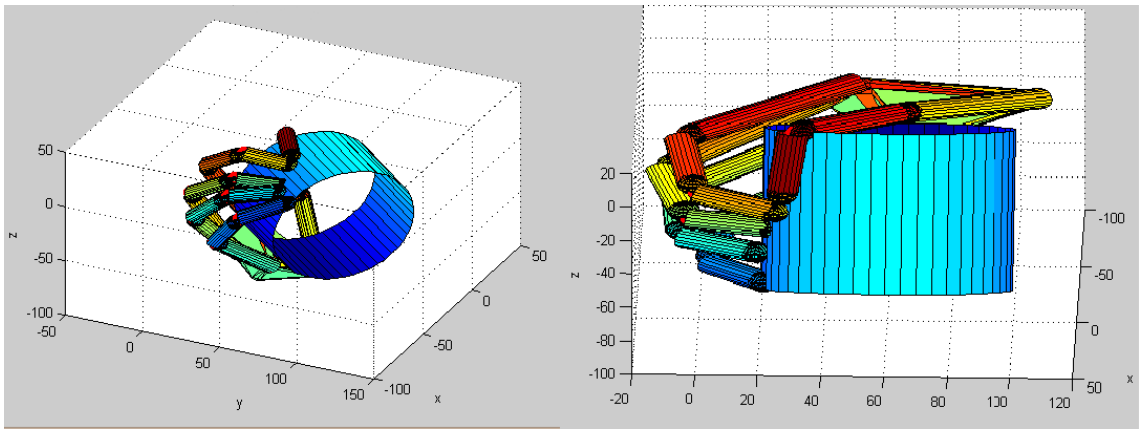

Figura 50. Simulación de agarre de un cilindro con la mano antropomórfica Gifu Hand III.

5.3.3 Cálculo de los puntos de contacto con un cubo.

Para calcular los puntos de contacto entre una mano y un cubo, SynGrasp posee su propia función, SGintSegCube. Esta función, es algo distinta que las anteriores. Primero, almacena las coordenadas de las caras del cubo en una matriz 3x24 (24: 6 caras con 4 puntos cada una, 3: las coordenadas x, y, z). Después, multiplica cada columna de la matriz, por la matriz inversa a la matriz de transformación, Htr, del cubo. Una vez creada la matriz, realiza una iteración de un total de 6 (como el número de caras). En cada iteración, almacena en una matriz, las coordenadas x, y, z de 3 puntos de la cara y evalúa la siguiente ecuación:

$$DEN = A \cdot (P1_x - P0_x) + B \cdot (P1_y - P0_y) + C \cdot (P1_z - P0_z) \quad (9)$$

donde:

- DEN: es un valor que si vale 0, indica que no hay puntos de contacto en esa iteración.
- P1: son las coordenadas del final del eslabón.
- P0: son las coordenadas del principio del eslabón.
- A: se obtiene de la siguiente ecuación:

$$A = y_1 \cdot (z_2 - z_3) - z_1 \cdot (y_2 - y_3) + (y_2 \cdot z_3 - y_3 \cdot z_2) \quad (10)$$

  - Y_1, Y_2, Y_3: son las coordenadas ‘y’ de tres puntos que forman la cara.
  - Z_1, Z_2, Z_3: son las coordenadas ‘z’ de tres puntos que forman la cara.

- B: es el resultado de:

$$B = x_1 \cdot (z_2 - z_3) - z_1 \cdot (x_2 - x_3) + (x_2 \cdot z_3 - x_3 \cdot z_2) \quad (11)$$

  - X_1, X_2, X_3: son las coordenadas ‘x’ de tres puntos que forman la cara.

- C: obtenido como:

$$C = x_1 \cdot (y_2 - y_3) - y_1 \cdot (x_2 - x_3) + (x_2 \cdot y_3 - x_3 \cdot y_2) \quad (12)$$

Si el valor DEN es distinto de 0, se evalúa la siguiente ecuación:

$$alphatemp = \frac{-(A \cdot P0_x + B \cdot P0_y + C \cdot P0_z + D)}{DEN} \quad (13)$$

donde:

- Alphatemp: el valor i-esimo de alpha.
- D: que se obtiene de:

$$D = x_1 \cdot (y_2 \cdot z_3 - y_3 \cdot z_2) - y_1 \cdot (x_2 \cdot z_3 - x_3 \cdot z_2) + z_1 (x_2 \cdot y_3 - x_3 \cdot y_2) \quad (14)$$

Con los 6 valores obtenidos de alphatemp, se rellena un vector llamado alphaVect y se evalúa cada uno, para comprobar, de la siguiente manera, si es un valor α válido:

- Obtiene el valor X_{temp}, Y_{temp} y Z_{temp} con las siguientes ecuaciones:

$$X_{temp} = P0_x + alphatemp \cdot (P1_x - P0_x) \qquad (15)$$

$$Y_{temp} = P0_y + alphatemp \cdot (P1_y - P0_y) \qquad (16)$$

$$Z_{temp} = P0_z + alphatemp \cdot (P1_z - P0_z) \qquad (17)$$

- Multiplica estos valores obtenidos, por la inversa de la matriz de transformación, Htr, del cubo.
- Si estas coordenadas entran dentro del cubo, se considera el alphatemp como α válido.

En las figuras 51 y 52, se aprecia el funcionamiento de la simulación de agarres de cubos con las manos ya proporcionadas por la herramienta. En la figura 52, al igual que en el caso del cilindro, se observa que la mano proporcionada por la propia herramienta, atraviesa varias partes del objeto. En la figura 53, se comprueba que la mano Gifu Hand III también funciona como las anteriores para este tipo de agarres, aunque también, presenta los problemas de la paradigmatic.

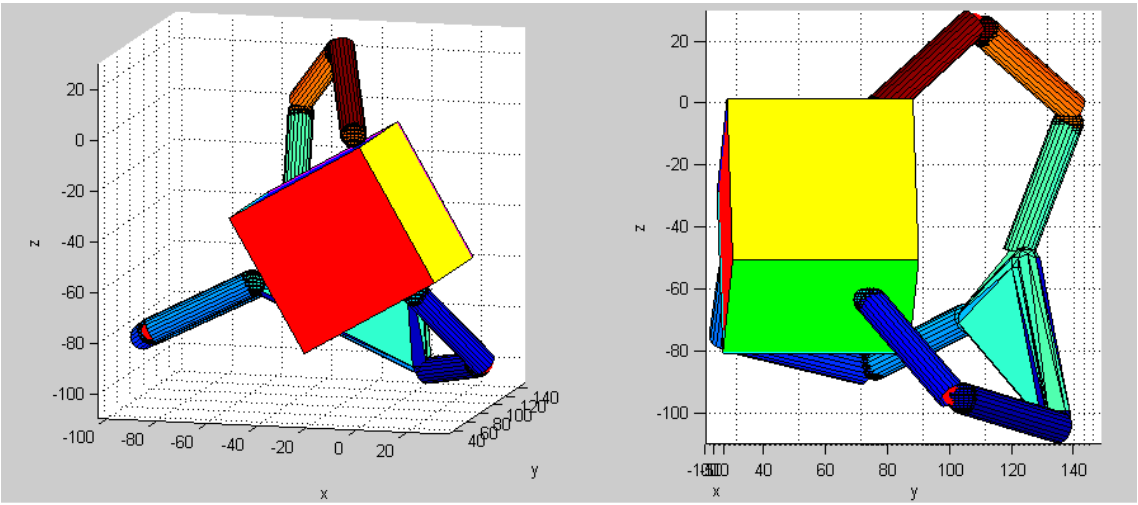

Figura 51. Simulación de agarre de un cubo con la mano no antropomórfica 3fingered.

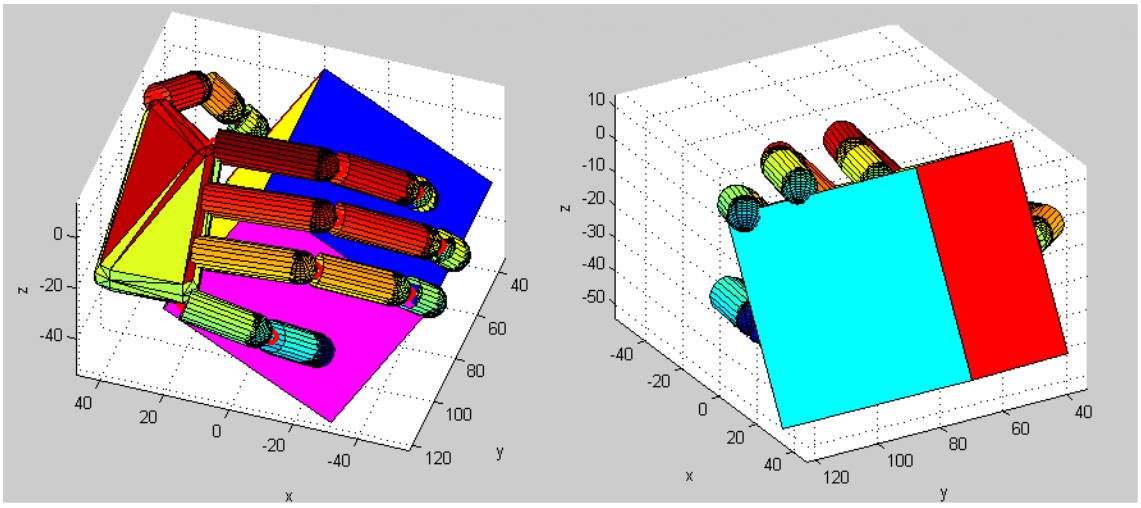

Figura 52. Simulación de agarre de un cubo con la mano antropomórfica paradigmatic.

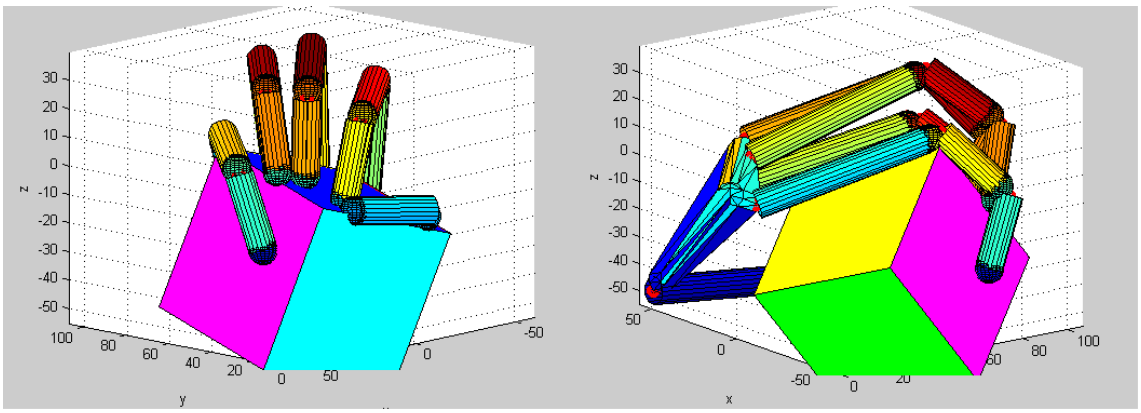

Figura 53. Simulación de agarre de un cilindro con la mano antropomórfica Gifu Hand III.

5.3.4 Cálculo de los puntos de contacto con objetos STL.

Para calcular los puntos de contacto entre una mano y un objeto STL, se ha creado la función, SGintSegstl. Ya que conocemos 2 puntos de cada eslabón de la mano, P_0 y P_1, correspondientes a las coordenadas de las articulaciones que encierran al eslabón, conocemos su longitud total, y por tanto, podemos dividir dicho eslabón en 4 partes o 5 puntos. Estas partes serán:

- El origen, al igual que en las funciones incluidas por SynGrasp, si el punto de contacto se encuentra aquí, α (valor entre 0 y 1) valdrá 0.
- El siguiente punto, corresponde al primer cuarto de eslabón, aquí α valdrá 0,25.
- En la mitad del eslabón, α valdrá 0,5.
- En el penúltimo punto, obtendremos un valor de α de 0,75.
- Y en el final del eslabón, α valdrá 1.

Con el fin de obtener el valor α que utiliza el resto del programa, se ha utilizado la ecuación correspondiente a la distancia euclídea entre 2 puntos.

Como conocemos todos los puntos del objeto stl, la función, se limita a recorrerlos, y evaluar la distancia entre el punto seleccionado y los puntos inicial y final del eslabón, para después, comparar los valores obtenidos (D1 y D0 de la figura 54) con otros almacenados, con el fin de comprobar si el punto evaluado está a 5 mm de cualquiera de los 5 puntos del eslabón. Se ha establecido una distancia mínima de 5 mm, ya que corresponde al radio del dedo de una mano. Cualquier punto que se encuentre a esa distancia mínima, o a menos del eslabón, se considerará que estará en contacto.

Se utiliza la distancia entre 2 puntos para limitar el rango de puntos válidos, a la zona comprendida entre el principio y el final del eslabón, evitando obtener falsos resultados de puntos que se encuentren a dicha distancia pero no estén comprendidos en la zona de trabajo del eslabón.

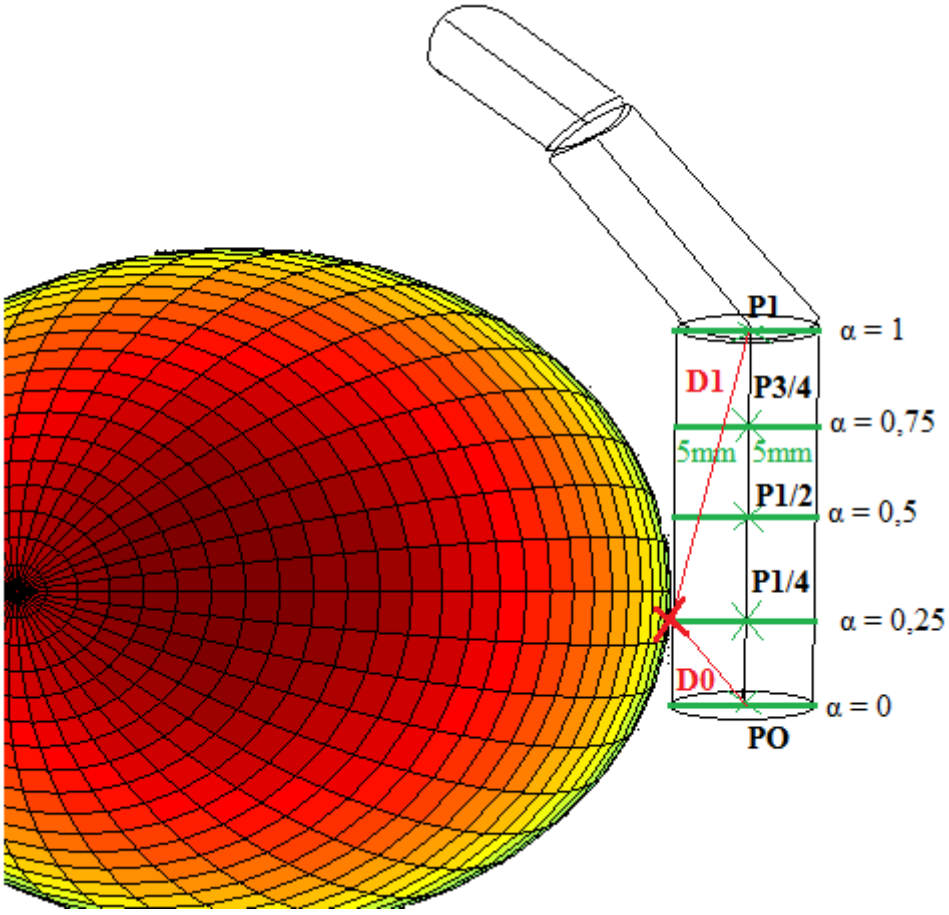

Figura 54. Comprobación de un punto válido.

Las ecuaciones utilizadas en estos cálculos son:

$$L = \sqrt{(P1_x - P0_x)^2 + (P1_y - P0_y)^2 + (P1_z - P0_z)^2} \quad (18)$$

donde:

- L: es la distancia entre el punto inicial y final del eslabón.

$$D1 = \sqrt{(P1_x - Pobj_x)^2 + (P1_y - Pobj_y)^2 + (P1_z - Pobj_z)^2} \quad (19)$$

$$D0 = \sqrt{(P0_x - Pobj_x)^2 + (P0_y - Pobj_y)^2 + (P0_z - Pobj_z)^2} \quad (20)$$

Donde:

- D1: Distancia desde el punto P1.
- D0: Distancia desde el punto P2.
- Pobj: Son las coordenadas del punto del objeto evaluado.

En las figuras 55 y 56, se aprecia el funcionamiento de la simulación de agarres de objetos STL, con las manos ya proporcionadas por la herramienta. Al igual que en los casos anteriores, la mano puede atravesar el objeto. En la figura 57, se comprueba que la mano Gifu Hand III también funciona con este tipo de agarres.

El principal problema de este método, es el tiempo de cálculo que requiere, ya que evalúa todos los puntos que forman un objeto STL (en el caso del objeto utilizado, unos 7000 puntos) en cada iteración y con cada eslabón, añadiéndole a esto, el problema incluido en SynGrasp, su método de agarre sólo consiste en cerrar la mano.

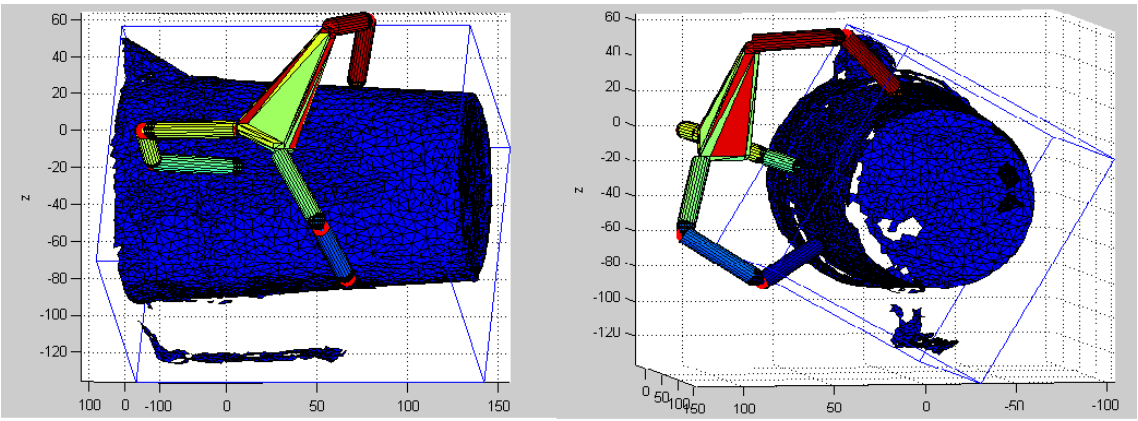

Figura 55. Simulación de agarre de un objeto STL con la mano no antropomórfica 3fingered.

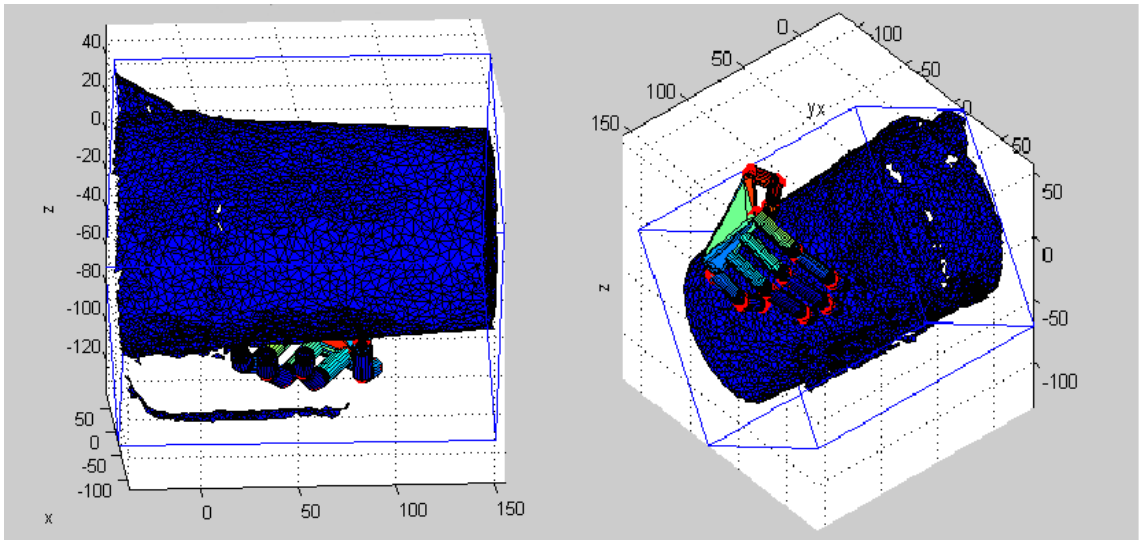

Figura 56. Simulación de agarre de un objeto STL con la mano antropomórfica paradigmatic.

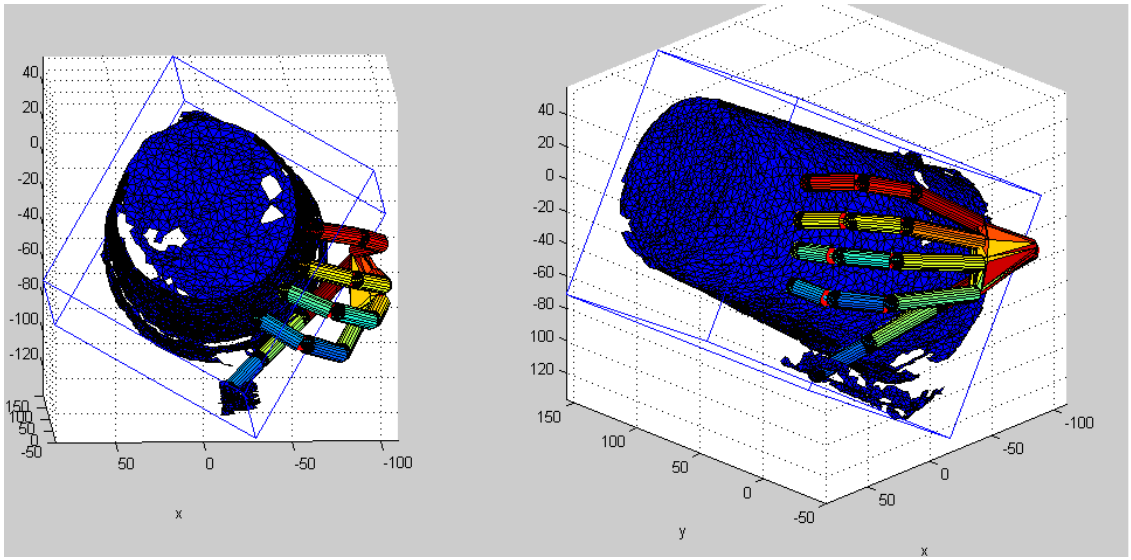

Figura 57. Simulación de agarre de un objeto STL con la mano antropomórfica Gifu Hand III.

5.3.5 Tipos de calidad de agarre.

Como ya hemos visto, SynGrasp incluye 5 tipos distintos de formas para evaluar la calidad del agarre entre la mano y el objeto:

- **Mev** (Manipulation Ellipsoide Volume): utilizando las matrices Jacobiana y de agarre de la mano, este método evalúa (teniendo en cuenta la configuración de la mano) el volumen elipsoidal de manipulación de ésta. Este volumen, se toma como medida de calidad de agarre. Para calcularlo, utiliza la siguiente ecuación:

$$Q = K_b \cdot \sqrt{Det(HO \cdot HO^T} = K_b \cdot (S1_{H0} \cdot S2_{H0} \dots Sn_{H0}) \quad (21)$$

  donde:
  - Q: es la calidad del agarre.
  - K_b: es un valor constante.
  - HO: es la matriz Jacobiana de la mano-objeto obtenida de:

$$HO = \left(G^T \cdot inv(G \cdot G^T)\right) \cdot J \qquad (22)$$

    donde G, es la matriz de agarre y J la matriz Jacobiana.

  - S_{H0}: son los valores singulares de la matriz H0.

  Debido a esto, la calidad es proporcional al producto de los valores singulares. Al maximizar el determinante, se maximiza también el volumen (calidad).

- **Gii** (Grasp Isotropy index): esta medida de calidad, está asociada a las posiciones de los puntos de contacto, utilizando medidas basadas en el álgebra de la matriz de agarre G. Este criterio, busca una distribución uniforme de las fuerzas de contacto en la torsión ejercida al objeto, es decir, intenta obtener un agarre isotrópico donde las magnitudes de las fuerzas internas sean similares. Para obtener este valor, se utiliza la siguiente ecuación:

$$Q = \frac{Smin_g}{Smax_g} \qquad (23)$$

  donde:
  - Q: es la calidad.
  - $Smin_g$: es el valor singular minimo de la matriz de agarre G.
  - $Smax_g$: es el valor singular máximo de la matriz de agarre G.

  Este valor, se aproxima a 1 cuando el agarre es isótropo (casi óptimo) y a 0 cuando el agarre se encuentra cerca de una configuración singular.

- **Msvg** (Minimun Singular Value of Grasp matrix): evalúa el mínimo valor singular de la matriz de agarre G, para obtener una medida de la calidad del agarre. Cuando un agarre está cerca de una configuración singular, al menos, un valor singular de la matriz de agarre vale 0. Por lo tanto, cuanto más alto sea este valor, mejor calidad tendrá el agarre.

- **Dtsc** (Distance To Singular Configurations): utiliza la configuración de la mano para evaluar la calidad del agarre. Al igual que en el caso anterior, busca el valor singular mínimo, pero esta vez, no es de la matriz de agarre G, sino de la matriz de agarre mano-objeto HO, ya que, un valor grande equivale a alejarse de configuraciones singulares en los dedos.

- **Uot** (Uniformity Of Transformation): este último criterio, también utiliza la configuración de la mano para obtener la calidad del agarre. Considera que, la transformación entre la velocidad de las articulaciones de los dedos y la velocidad del objeto, es uniforme cuando la contribución de cada articulación, es la misma en todos los componentes de la velocidad del objeto. Para obtener esta medida, se sigue la siguiente ecuación:

$$Q = \frac{Smax_{H0}}{Smin_{HO}} \qquad (24)$$

donde:

- Q: es la calidad.
- $Smin_{H0}$: es el valor singular mínimo de la matriz de agarre mano-objeto H0.
- $Smax_{H0}$: es el valor singular máximo de la matriz de agarre mano-objeto H0.

La calidad del agarre, será mejor cuanto más se aproxime Q a 1.

En los cálculos obtenidos para este proyecto, se ha utilizado el tipo de calidad de agarre mev.

6 Presupuesto.

Como este trabajo de fin de grado es de investigación y está basado en la utilización y mejora de una herramienta de investigación, no tiene mucho sentido hablar de presupuesto, ya que la *toolbox* es gratuita, la licencia de Matlab para estudiantes es gratuita y pese a simular la mano Gifu Hand III, ésta no se ha utilizado y por tanto no es necesaria.

Aun así, se podría realizar un presupuesto, contabilizando las horas de trabajo, tanto del autor, como del tutor y el valor de los equipos utilizados, siendo el presupuesto el siguiente:

Tabla 5. Presupuesto personal.

ACTIVIDAD	*EUROS/HORA*	*Nº HORAS*	*EUROS*
Investigación	20	90	1800
Programación y pruebas	15	110	1650
Escritura	10	75	750
Reuniones con tutor y corrección	60	20	1200
Total		295	5400

Tabla 6. Presupuesto material.

ARTÍCULO	*CANTIDAD*	*PRECIO*
Ordenador portátil MSI GP60 2PE-417-XES con SO Windows 8.	1	740

Tabla 7. Presupuesto final.

TIPO DE COSTE	*PRECIO*
Coste personal	5400
Coste material	740
Subtotal	6140
I.V.A. (21%)	1289,4
TOTAL	**7429,4**

De todas formas, si este proyecto fuese realizado por una empresa, y para una futura implementación en la Gifu Hand III si se podría añadir a este presupuesto el precio de la mano y de la licencia de Matlab.

7 Conclusiones y futuras líneas de trabajo.

Como hemos comprobado, hemos conseguido mejorar en gran medida la herramienta SynGrasp, con la adición de un nuevo modelo de mano y un nuevo tipo de objeto. Este nuevo tipo de objetos, consigue generalizar la herramienta en gran medida, ya que, permite incluir cualquier tipo de objeto, sea cual sea su geometría, con tal de estar en formato STL, permitiendo trabajar a la herramienta con objetos mucho más complejos que los incluidos en ella. Además, se han añadido, corregido y optimizado funciones para mejorar el funcionamiento de esta *toolbox*.

Pese a ello, esta herramienta sigue presentando varios problemas, como hemos podido ver a la hora de realizar el agarre de objetos (manos atravesando objetos). Por lo que, una de las posibles líneas de trabajo futuro, puede ser, optimizar y resolver este tipo de problemas de la herramienta, así como implementar un nuevo algoritmo de agarre (ya que el algoritmo actual puede mejorarse bastante), o mejorar los métodos de calidad (actualmente se basan en la posición de la mano respecto a sus puntos de singularidad, pero no a si el objeto está bien cogido o si el agarre es real y no solo se está tocando el objeto).

Otra posible mejora que se puede efectuar, es optimizar la implementación de objetos STL, para reducir su tiempo de cálculo (actualmente, una iteración de un objeto STL, dura 7-10 minutos), evaluando solo los puntos del contorno de dichos objetos y reduciendo el número de estos (si están muy próximos entre sí), con el fin de obtener datos e incluirlos en una mano que trabaje a tiempo real. Una posible aplicación de esta última mejora, sería la utilización conjunta de una mano robótica y una Kinect, en un robot humanoide, para evaluar objetos y cogerlos en tiempo real.

8 Bibliografía.

[1] http://www.monografias.com/trabajos88/evolucion-inteligencia-artificial-aplicada-robotica/evolucion-inteligencia-artificial-aplicada-robotica.shtml última visita Agosto 2014.

[2] http://www.ehow.com/video_4989818_invented-robotic-hand.html última visita Agosto 2014.

[3] http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.4.htm última visita Agosto 2014.

[4] Evolution of Robotic Hands, H.D. Bos, M. Wassink. University of Twente.

[5] CABAS, Ramiro, "Metodología de diseño de manos robóticas basada en los estados de su sistema accionador". Director: Dr. Carlos Balaguer Bernaldo de Quiros. Universidad Carlos III de Madrid, Departamento de Sistemas y Automática, 2011.

[6] GRIFANTINI, Kristina. "Un agarre robótico más simple y delicado". MIT technology review. Publicado por Opinno. 2009.

[7] ZUÑIGA TENESACA, Daniel Alejandro, ANDRADE ZEAS, Diego Miguel. "Diseño y construcción de una mano robótica para la enseñanza del alfabeto dactilológico universal para personas sordomudas". INGENIUS Nº6 (julio/diciembre). pp. 67-84 .ISSN: 1390-650X.

[8] SMITH, Wallace. "La maravillosa mano humana". El mundo del mañana. 2014.

[9] Apuntes de la asignatura "Robótica industrial". Universidad Carlos III.

[10] Apuntes de Cinemática. Universidad de Zaragoza, disponibles en: http://wzar.unizar.es/acad/cinesio/Documentos/Pulgar_Apuntes_2010.pdf última visita Agosto 2014.

[11] http://biomecanicadelmiembrosuperior.blogspot.com.es/ última visita Agosto 2014.

[12] http://es.slideshare.net/juangonzalezleija/exploracin-de-mueca última visita Agosto 2014.

[13] http://introduccionalapm.blogspot.com.es/2011/05/articulacion-de-manos-y-dedos.html última visita Agosto 2014.

[14] SICILIANO, Bruno, KHATIB, Oussama. Handbook of robotics. Springer 2008, chapter 15.

[15] PIZARRO DE LA HOZ, Vladimir, RINCÓN SANCHEZ, Giovanny Alberto. "Diseño e implementación de mecanismo de prensión para mano robot antropomórfica". Director: Dr. Oscar Fernando Avilés Sánchez. Universidad militar Nueva Granada, Bogotá 2012.

[16] GONZÁLEZ VILLELA, Víctor Javier, ARNEZ PANIAGUA, Víctor Augusto, HERNÁNDEZ AVILÉS, Ernesto. "Diseño mecánico de mano antropomórfica orientada a sujeción de cuerpos amorfos". XVI congreso internacional de la SOMIN. 2010.

[17] Teoría de actuadores. Asociación de la industria eléctrica electrónica, Chile. Disponible en: http://www.aie.cl/files/file/comites/ca/abc/actuadores.pdf última visita Agosto 2014.

[18] FLORES PORTILLA, Edgar, PIÑA QUINTERO, Roberto, AVILÉS SÁNCHEZ, Oscar, NIÑO SUAREZ, Paola. "Diseño del mecanismo actuador de un dedo robot antropomórfico". Revista Facultad de Ingeniería Universidad de Antioquia. 2011.

[19] Pei Q., Ha S.M., Pelrine R., Kovacs G., 2007, Enhanced performance of IPN electroelastomer, SPIE: Smart Materials and Structures, San Diego.

[20] http://www.wisegeek.com/what-is-a-double-acting-actuator.htm#didyouknowout última visita Agosto 2014.

[21] Shadow Robot Company© Air muscles Datasheets. Disponible en: http://robots-argentina.com.ar/Actuadores_manos.htm última visita septiembre 2014.

[22] Folleto de músculos neumáticos de la empresa FESTO©. Disponible en: http://www.festo.com/cms/es_corp/9790.htm última visita Septiembre 2014.

[23] http://www.arqhys.com/contenidos/musculos-artificiales-neumaticos.html última visita Septiembre 2014.

[24] http://prezi.com/oxlo5zgsjdol/sensores-tactiles/ última visita Agosto 2014.

[25] http://mindtrans.narod.ru/hands/hands.htm última visita Agosto 2014.

[26] QUINAYÁS BURGOS, Cesar Augusto. "Diseño y construcción de una protesis de mano funcional adaptada a varios agarres". Director: Dr. Oscar Andrés Vivas Albán. Universidad del Cauca. 2010.

[27]Manual de especificaciones técnicas de la mano Shadow dexterous hand. Shadow Robot Company©, disponible en: http://mindtrans.narod.ru/pdfs/shadow_dexterous_hand_technicalspecification_E1_20130101.pdf última visita Agosto 2014.

[28] EH1 Milano series extrinsic robotic hand basic user guide. Prensilia©. Disponible en: http://www.prensilia.com/files/support/doc/Prensilia%20EH1%20basic_10.pdf última visita Agosto 2014.

[29] Especificaciones técnicas Sheffield arm. Elumotion©. Disponible en: http://www.davidbuckley.net/RS/HandResearch.htm última visita Agosto 2014.

[30] Especificaciones técnicas Elu hand 2. Elumotion©. Disponible en: http://mindtrans.narod.ru/pdfs/Elu2-Hand-Data-Sheet-24-2-10-a.pdf última visita Agosto 2014.

[31] Novedades Técnicas Elu Hand 2 Elumotion©. Disponible en: http://www.elumotion.com/Pdfs/Elu2-Hand-PR-19-4-10.pdf última visita Agosto 2014.

[32] ficha técnica Robonaut 1. NASA. Disponible en: http://robonaut.jsc.nasa.gov/R1/sub/hands.asp última visita Agosto 2014.

[33] http://www.elmundo.es/ciencia/2014/07/24/53c90b4b268e3e566b8b4583.html última visita Agosto 2014.

[34] SUAREZ, Raúl, GROSCH, Patrick. "Mano mecánica MA-I". Universidad politécnica de Cataluña.

[35] DLR hand 2. DLR disponible en: http://www.dlr.de/rmc/rm/en/desktopdefault.aspx/tabid-3802/6102_read-8923/ última visita Agosto 2014.

[36] Barret Hand overview. Barret Thecnology ©Inc.

[37] Barret hand GUI manual. Barret Thecnology©Inc. Disponible en http://web.barrett.com/support/BarrettHand_Documentation/BH_Control_GUI_Manual.pdf última visita Agosto 2014.

[38] Listado de clientes Robot Hand, disponible en: http://robothand.eu/en/clients/ última visita Agosto 2014.

[39] Kawasaky and Moury Laboratories. GIFU University. http://robo.mech.gifu-u.ac.jp/index_e.html última visita Agosto 2014.

[40] http://robothand.eu/en/products/robotic_hands/robot_hand_gifu_hand_iii/ última visita Agosto 2014.

[41] MOURI, Tetsuya, KAWASAKI, Haruhisa. "A novel anthropomorphic robot hand and its master slave system". Human robots: Human-like Machines. Edited by Mathias Hackel. Pp 642. 2007.

[42] NAVA RODRÍGUEZ, Nestor Eduardo. "Advanced Mechanics in robot systems". Springer. 2011.

[43] KAWASAKI, Haruhisa, SHIMOMURA, Hisayuki, SHIMIZU Yuuji. "Educational-industrial complex development of an anthropomorphic robot hand 'Gifu hand'". Advanced Robotics, Vol 15, No 3, pp 357-363. 2001.

[44] KAWASAKI, Haruhisa, KOMATSU, Tsuneo. "Mechanism design of anthropomorphic robot hand: Gifu Hand I". Journal of Robotics and Mechatronic, Vol 11, No 4, pp 269-273, 1999.

[45] KAWASAKI, Haruhisa, KOMATSU, Tsuneo, UCHIYAMA, Kazunao. "Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu Hand II". IEEE/ASME transactions on mechatronics, Vol 7, No 3. 2002.

[46] MOURI, Tetsuya, KAWASAKI, Haruhisa, YOSHIKAWA, Keisuke, TAKAI, Jun, ITO, Sathosi. ”Anthropomorphic robot hand: Gifu Hand III”. ICCAS. 2002.

[47] Andrew Miller and Peter K. Allen. "Graspit!: A Versatile Simulator for Robotic Grasping". IEEE Robotics and Automation Magazine, V. 11, No.4, Dec. 2004, pp. 110-122.

[48] http://openrave.org/docs/latest_stable/openravepy/databases.grasping/ última visita Septiembre 2014.

[49] M. Malvezzi, G. Gioioso, G. Salvietti, D. Prattichizzo, A. Bicchi. SynGrasp: a MATLAB Toolbox for Grasp Analysis of Human and Robotic Hands. In Proc. IEEE Int. Conf. on Robotics and Automation, To appear, Karlsruhe, Germany, 2013

[50] Robotics, Vision and Control fundamental algorithms in matlab, Peter Corke.

[51] Chua, C. K; Leong, K. F.; Lim, C. S. (2003), Rapid Prototyping: Principles and Applications (2nd ed.), World Scientific Publishing Co, ISBN 981-238-117-1 Chapter 6, Rapid Prototyping Formats. Page 237, "The STL (STeroLithography) file, as the de facto standard, has been used in many, if not all, rapid prototyping systems." Section 6.2 STL File Problems. Section 6.4 STL File Repair.